最新実用カラー版

詳解 HTML & Java Script 辞典

Up-to-date utility color edition
full explained
HTML and Java Script
dictionary

岡蔵龍一　半場方人　著

SHUWA SYSTEM CO.,LTD.

最新実用カラー版

詳解 HTML & JavaScript 辞典

Up-to-date utility color edition
full explained
HTML and Java Script
dictionary

岡蔵龍一　半場方人　著

SHUWA SYSTEM CO.,LTD.

はじめに

早いもので、HTML4.0が発表されてからすでに2年が経過しました。現在では、すべてではありませんが、その機能も多く利用できるようになり、その役割や目的も広く理解されるようになってきています。本書のＨＴＭＬパートでは、初心者でもHTML4.0を正しく理解し利用できるよう、実際に利用可能な部分を中心にして、具体例と共に簡潔にまとめてみました。また、現実的に考えてHTML4.0の仕様外でも利用されることが想定されるものについては、そのことを説明した上で解説しています。

本書では、HTMLと密接に関係しているスタイルシートに関しても、その解説をHTMLパートに含めることにしました。HTMLではなくスタイルシートで行うことが望ましいものについては、関連ページとして示していますので、ぜひ挑戦してみてください。また、より多くの環境で利用できるWebページを作成するためのアクセシビリティに関するヒントも多く掲載しています。

本書の内容を総合的に利用することによって、より正しく優れたWebページの制作が可能になることでしょう。

岡蔵　龍一

「詳解HTML&JavaScript辞典」が出版されてから約2年、JavaScriptはますます一般的になり、Webページの表現手段の1つとして、ごく普通に使われるようになりました。そして、当時と比べるとJavaScriptのバージョンも上がり、その能力はますます高く、複雑な処理もこなせるようになっています。しかしながら、それに合わせてどのブラウザのどのバージョンがどのJavaScriptをサポートしているかが、ますます分かりにくくなってきていることも確かです。

本書では、前回から追加されたり、新たに仕様が公開されたJavaScriptを収録することはもちろん、それぞれのスクリプトが、JavaScriptをサポートしている2大ブラウザであるNetscape NavigatorとInternet Explorerのどのバージョンに対応しているか、一目で分かるようにしています。

ぜひお気に入りのJavaScriptを見つけて、個性的なWebページ作りにお役立てください。

半場　方人

本書の使い方

対応ブラウザ

Netscape Navigator2.0/3.0/4.0(JavaScriptは4.06まで)と、Internet Explorer3.0/4.0/5.0で動作確認をした結果を表示しています。OSによる対応・未対応は「解説」をお読みください。なお、文字が白くなっている場合は、属性の一部が使用できません。

書式

タグやスクリプトの書式。イタリック文字は、数値などの値を設定する部分です。HTMLパートでは、下の表で値の解説をする場合があります。

解説

タグやスクリプトの使用方法や注意点などを、文章で分かりやすく説明します。

Sample

改変可能なHTML・JavaScriptのソースを掲載しています。また、サポートページから、コピーして使用することもできます。

柱

本書の項目は、HTMLパートはタグの機能ごと、JavaScriptはオブジェクト名ごとに分類されています。その項目名で探す時の目安になります。

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

HTML

画像への回り込みを解除する

<BR clear="*どちら側の画像に対して解除するか*">

【どちら側の画像に対して解除するか】

left	左側の画像に対する回り込みを解除
right	右側の画像に対する回り込みを解除
all	両側の画像に対する回り込みを解除

Internet Explorer

IMG要素に対して「align="left"」を指定すると、このように画像が左側に配置されて、それに続くテキストが右側に表示されます。

この回り込みを解除するためには、BR要素のclear属性を使用します。

Netscape Navigator

IMG要素に対して「align="left"」を指定すると、このように画像が左側に配置されて、それに続くテキストが右側に表示されます。

この回り込みを解除するためには、BR要素のclear属性を使用します。

解説 画像を左、又は右に配置してテキストをその横に回り込ませた場合の、回り込みを解除します。
ただし、HTMLで回り込みを制御するための属性は廃止予定となっていますので、スタイルシートを使った方がよいでしょう。

Sample

```
<P>
<IMG src="../images/fish.gif" width="230" height="145"
alt="サンプル画像" align="left">
IMG要素に対して「align="left"」を指定すると、このように画像が左側に配置されて、
それに続くテキストが右側に表示されます。
<BR clear="left">
</P>

<P>
この回り込みを解除するためには、BR要素のclear属性を使用します。
</P>
```

「画像とマルチメディア」の「画像にテキストを回り込ませる - 画像の位置指定 -」:P.128参照
「スタイルシート」の「回り込みを解除する」:P.231参照

130 画像とマルチメディア

画像を配置する

注意 重要 Tip コラム ページの項目や、その前後の項目に関連した注意点やワンランク上のテクニックなどを紹介しています。

見出し

ページ上でできること・したいことから、タグやスクリプトを探すことができます。

JavaScript

レイヤーの背景の色を変える

bgColor 【プロパティ】

NN4.0
NN4.06

表示例の解説。ブラウザ名か、サンプルの動きを表しています。本書では、注記がない場合は、Internet Explorer5.x の画面を掲載しています。

Before

*レイヤーのバックの色を変える

ここにマウスを持ってくるとバックの色が変わります...

After

*レイヤーのバックの色を変える

ここにマウスを持ってくるとバックの色が変わります...

サンプルソースのブラウザ表示例。タグやスクリプトの効果が一目瞭然です。

解説

<LAYER>タグ内にJavaScriptを記述することにより、その階層のレイヤーのみに影響するJavaScriptを設定することができます。
サンプルでは、レイヤー上にマウスカーソルを乗せたり、レイヤーからマウスカーソルが離れた時、イベントが発生してレイヤーのバックの色が変わります。

ツメ

分類されたタグやスクリプトの機能を一言で表しています。機能ごとに探す時に便利です。

Sample

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*レイヤーのバックの色を変える<P>
<LAYER NAME="layer1" VISIBILITY="show" BGCOLOR="blue" 
TOP=50 LEFT=100 WIDTH=200 HEIGHT=200>
    <LAYER NAME="layer2" BGCOLOR="blue" TOP=50 LEFT=50 
WIDTH=100 HEIGHT=100 onMouseOver="changeColor('red')"on
MouseOut="changeColor('blue')">
        <P>ここにマウスを持ってくるとバックの色が変わります...</P>
        <SCRIPT LANGUAGE="JavaScript1.2">
        <!--
        function changeColor(NC) { bgColor=NC }
        //-->
        </SCRIPT>
    </LAYER>
</LAYER>
</BODY>
</HTML>
```

改行せずに1行で書かなければいけないスクリプトを表しています。

JavaScript

レイヤーを操作する

<LAYER>→コラム「Netscape Navigator4.xのレイヤーについて:1〜4」:P.204参照

Layer オブジェクト 427

リンク

文法的に関連がある部分や、応用する場合に必要となる箇所を参照することができます。

目次

HTMLパート

HTMLについて

ページの基礎となる内容

テキスト

スタイルとレイアウト

リンク

リスト

テーブル

画像とマルチメディア

入力フォーム

フレーム

スクリプト

スタイルシート

アクセシビリティ

HTML付録

JavaScriptパート

JavaScriptについて

ナビゲーターオブジェクト

navigatorオブジェクト

screenオブジェクト

eventオブジェクト

windowオブジェクト

frameオブジェクト

documentオブジェクト

historyオブジェクト

locationオブジェクト

Linkオブジェクト・Anchorオブジェクト

Formオブジェクト

Areaオブジェクト

Imageオブジェクト

Layerオブジェクト(Netscape Navigator)

スタイルシート(Internet Explorer)

ビルトインオブジェクト

Dateオブジェクト

Mathオブジェクト

stringオブジェクト

Arrayオブジェクト

functionオブジェクト

Objectオブジェクト

Booleanオブジェクト

Numberオブジェクト

その他

複数のオブジェクトで使用できるプロパティ・メソッド

ビルトイン関数(top-level関数)

リファレンス

JavaScript付録

索引パート

巻末付録

詳解
HTML
&
Java Script
辞典

HTML

1. HTML4.0とは？

インターネットのブームは、Mosaicという画像を表示できるWebブラウザと共に突然やってきました。

そして、それに続いてNetscape NavigatorというWebブラウザが登場しました。Netscape Navigatorは、プラグイン機能で様々なデータの取り扱いを可能にし、フレーム機能やJava Scriptなどのスクリプト言語も利用できるようにしていきました。

そのうち、途中から登場してきたInternet ExplorerとNetscape Navigatorは、激しいシェア争いのために、次々と独自の機能を追加するようになっていきます。そして、ユーザーは、「新しい機能」をこぞって使うという時期がしばらく続きました。当時は単純にアクセス数が多ければ多いほどよいと思われていた時代で、新しい機能を使って「こんなこともできる!」というページを作るだけで、アクセス数が多くなったことを記憶しています。

しかし、しばらくすると、それによる弊害も現れてきました。常に最新のWebブラウザを利用していないと、見ることのできないページが多くなってしまったのです。当時のWebサイトには、トップページに「○○○のバージョンX.XX以上でご覧ください」などと書かれていたことがそれを象徴しています。

本来は多くの人に見て欲しいはずのWebページが、ユーザーを限定してしまう方向へと進んでしまったわけです。

ちょうどその頃、WWWの標準化を行っているW3Cが、HTML4.0という仕様を発表しました。この仕様は一言でいえば、「今までのHTMLから見栄えに関するものを取り除き、HTMLでは純粋に文書の構造のみを表して、見栄えやレイアウトにはスタイルシートを使おう」というものでした。

もちろん、突然そのような方向へ切り替えようといっても、Webブラウザ側が対応していません。そのような理由で、HTML4.0には純粋なHTML4.0の他に、今まで使われていたタグも利用できる移行期間用のHTML4.0も用意されました。

では、見栄えなどの表示に関する部分をHTMLから切り離して、HTMLでは文書の構造のみを表すとどうなるのでしょうか?

ここでいう「文書の構造を表す」とは、特に難しい意味ではなく、「大見出し」・「普通の段落」・「強調部分」などの文書中での意味によってタグを付けていくということです。

つまり、文書の構成要素を「大きな文字」ではなく「大見出し」、「太字」ではなく「強調部分」などとして考えようということです。

ところで、皆さんがワープロなどで文書を作成する時に、「大見出し」はどのようにレイアウトしているでしょうか？

大抵の場合は、ゴシック系の大きな文字などで表すと思うのですが、実際にはそれがA3の紙に出力するのか、ハガキに出力するのか、カラーで出力するのかなどによって、同じ「大見出し」でもサイズなどが違ってくると思います。同じ見出しでも、最終的な出力対象によって表現は様々なのが普通だということです。

これをHTMLに当てはめてみると、文書構造のみを表していれば、様々な出力媒体で利用できるようになるということが分かります。

出力側でその構成要素に合わせた表現にして出力ができれば、普通の標準的なWebブラウザはもちろん、Lynxなどのテキスト専用ブラウザ、携帯用の端末や電話、音声出力、点字などでも利用できるようになるということです。これがもし、単に「24ポイントの文字」や「太字」などであれば、そのまま表現するしか方法はありません。

HTML4.0では、上記のように様々な出力媒体への対応を考慮すると同時に、様々な状況の人がより簡単に情報にアクセスできるような機能も盛り込まれました。

例えば、マウスが使えない状況を想定してタブキーやショートカットキーも設定できるようになっていますし、ある形式のデータが利用できない人のために、その情報を伝えるための別の方法を提供する機能も備えています。

しかし、現在それらの機能が徐々に利用できるようになってきているとはいえ、すべてのWebブラウザがこれらの機能を満たしているわけではありません。表示方法を制御するはずのスタイルシートも、なかなか完全にはサポートされていないのが現状です。

そのような意味では、HTML4.0を使うと都合の悪い部分も出てくることは確かですが、「移行期間用」のHTML4.0も用意されています。より多くの人のアクセスを可能にするHTML4.0を今から利用し、その意味を理解しておくことは、決して無駄なことにはならないはずです。

※本書では、この「移行期間用」のHTML4.0の機能の中で、現在でも利用可能なものを中心に解説しています。HTML4.0のより詳しい内容については、同シリーズの「詳解HTML4.0&Cascading Style Sheet辞典」を参照してください。

※本書では、一部でテキストブラウザの代表である、「Lynx」での表示例も掲載しています。

HTML

HTMLについて

2. 要素と属性

ここでは、HTMLで使われる用語について説明します。
まず、次の例を見てください。

```
HTMLの用語について

HTMLで使われる用語を正しく知っておくと、文法や
その構造がいっそう理解しやすくなります。・・・
```

上の例では、まず、「HTMLの用語について」という見出しがあります。そして、その内容について書かれた段落が続いています。
これは、人間は見れば分かることなのですが、コンピュータがそれを理解するのは簡単なことではありません。上の文章には、その構成要素として「見出し」と「段落」の2つがあります。HTMLでは、この構成要素をその名前を表すタグで囲うことによって、明確に示すようにしています。
つまり、構成要素の範囲を表すために開始タグと終了タグでその区切りを示し、その構成要素の種類を示すための要素名と、その構成要素の属性を指定して使用するのです。

```
<H1>HTMLの用語について</H1>

<P>
HTMLで使われる用語を正しく知っておくと、文法や
その構造がいっそう理解しやすくなります。・・・
</P>
```

このように、文書の構成要素にタグを付けることによって、コンピュータはそれが何であるかを簡単に理解できるようになります。
ここで、見出しをセンタリングしたい場合、その開始タグに構成要素の属性を指定することができます(ここでは、属性を簡単な例で説明するために「align属性」を使用していますが、本来はこのような場合には、スタイルシートを使うことが推奨されています)。これは、あくまで構成要素の内容に対する属性なのですが、それを示す場所として開始タグを利用することになっています。

```
<H1 align="center">HTMLの用語について</H1>

<P>
HTMLで使われる用語を正しく知っておくと、文法や
その構造がいっそう理解しやすくなります。・・・
</P>
```

したがって、上記の例の見出し部分で考えると、それぞれの用語は次の部分を指すことになります。

・**要素(タグも含めた構成要素全体)**

```
<H1 align="center">HTMLの用語について</H1>
```

・**タグ**

```
<H1 align="center"> </H1>
```

・**要素名**

```
H1
```

・**属性(構成要素に対する属性)**

```
align="center"
```

・**要素内容**

```
HTMLの用語について
```

※属性が複数ある場合、スペースで区切って順不同で指定できます。また、属性の値は基本的に「"」で囲って示すようにしてください。

※HTML4.0では、要素名や属性を書く場合は、大文字でも小文字でもよいことになっています。本書のHTMLパートでは、要素名と属性とを明確に区別するために、要素名は大文字で、属性は小文字で表記しています。

3. ブロックレベルとインライン

文書中で使用される要素の多くは、ブロックレベル要素とインライン要素に分類することができます。一般に、ブロックレベル要素は、他のブロックレベル要素やインライン要素を含むことができます。

また、インライン要素の中には、置き換え要素と呼ばれるものがあります。

・ブロックレベル要素

その名の通りブロック(ひとつのかたまり)として表される要素で、見出し・段落・リスト・表などが該当します。一般的に、その前後は改行されます。

ADDRESS	BLOCKQUOTE	CENTER	DIV	DL
FIELDSET	FORM	H1～H6	HR	MENU
NOFRAMES	NOSCRIPT	OL	P	PRE
TABLE	UL			

・インライン要素

行の一部として含まれる要素を指し、リンク・強調・略語などの部分が該当します。

A	ABBR	ACRONYM	APPLET	B
BASEFONT	BIG	BR	BUTTON	CITE
CODE	EM	FONT	I	IFRAME
IMG	INPUT	KBD	LABEL	MAP
OBJECT	Q	S	SAMP	SCRIPT
SELECT	SMALL	SPAN	STRIKE	STRONG
SUB	SUP	TEXTAREA	TT	U
VAR				

・置き換え要素

表示される時に、その要素自体があるもので置き換わるような要素を指します。

IMG	INPUT	TEXTAREA	SELECT	OBJECT

4. 色の指定方法

HTMLで色を指定するには、2通りの方法があります。ただし、色を指定する場合はHTMLを使用せずに、スタイルシートを利用することが推奨されています。

実際の色とその値については、巻末付録「カラーチャート1〜3」を利用すると便利です。

・16進数で指定する

「#」記号に続けて、RGB(Red, Green, Blue)の各値を2桁ずつの16進数で示します。例えば、赤を指定する場合には「#FF0000」となります。

・色の名前で指定する

HTML4.0では、基本的な16色(巻末付録「カラーチャート1：HTML4.0で名前が定義されている色」参照)については、色を名前で指定することができます。例えば、赤の場合は「red」を指定できます。この場合、大文字と小文字は区別されません。

5. ファイルの位置指定 - 絶対URLと相対URL -

例えば、ある部分をリンクさせる場合はリンク先のHTMLファイルの位置を、画像を表示させたい場合にはその画像の位置を指定する必要があります。HTMLでは、この位置をURLで示すのですが、それには2通りの方法があります。

・絶対URL

これは、Webブラウザでページを見ている時にアドレスバーなどに表示されている、「http://」で始まる形式の指定方法です。

この方法で指定すると、そのデータの転送方式やサーバー、サーバー内での位置まで完全に指定することになります。

一般に、自分のサイト内から他のサイトへとリンクする場合など、他のサイトのファイルに対して使用される形式です。

【例】 `http://www.w3.org/`

・相対URL

相対URLは、同じサイト内で参照を行う場合など、同じディスク上のファイルを参照する場合に利用される形式です。

この場合、現在のファイルの位置を基準として、ディレクトリ(フォルダ)の階層の上下を表すことによって位置を示します。自分でホームページを作成している場合など、オフラインの状態でも利用できるようにするためには、この方法で指定してください。

また、サイト内のデータをまるごと他のサーバーに移し変える場合なども、相対URLにしておけば、そのまま移行することができます(絶対URLにした場合は、修正する必要が出てきます)。

相対URLの指定方法は、自分より下の階層にあるファイルの場合は、そのディレクトリ名からファイル名までを順に「/」で区切って記述していきます。上の階層を示すには、1つ上を示すごとに「../」を付けて指定します。

・同じ階層(ディレクトリ)のファイルを示す場合：

```
ファイル名
```

・1つ下の階層(同じ階層にあるディレクトリ内)のファイルを示す場合：

```
ディレクトリ名/ファイル名
```

・2つ下の階層のファイルを示す場合：

```
ディレクトリ名/ディレクトリ名/ファイル名
```

・1つ上の階層のファイルを示す場合：

```
../ファイル名
```

・2つ上の階層のファイルを示す場合：

```
../../ファイル名
```

・1つ上の階層の別ディレクトリのファイルを示す場合：

```
../ディレクトリ名/ファイル名
```

HTML

どのバージョンのHTMLを使っているかを示す

<!DOCTYPE HTML ～>

解説

そのファイルが、どのバージョンのHTMLで書かれているかを示します。
HTMLには、HTML2.0・HTML3.2・HTML4.0などのバージョンがあり、更にHTML4.0は3種類に分けられます。使用するバージョンによってお決まりの書き方がありますので、そのまま文書の先頭に入れてください。ここで使用するバージョンを示したら、そのバージョンの決まりに従ったHTMLを書く必要があります。
本書の内容に沿ったページを作成するのであれば、以下のサンプルに示すようにHTML4.0のTransitional DTDを指定すればよいでしょう。
なお、<!DOCTYPE>は要素を表すタグではなく、SGMLの「文書型宣言」です。これを記述することによって、その文書内で使用するDTD(Document Type Definition：文書型定義)を宣言します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>タイトル</TITLE>
</HEAD>
<BODY>

</BODY>
</HTML>
```

lang="ja" →「アクセシビリティ」の「何語で書かれているかを指定する」：P.242参照

コラム

HTMLのバージョンと<!DOCTYPE>の書き方

以下に、現在使用されていると思われる主なHTMLの種類(バージョン)と、それを使用した場合の<!DOCTYPE>の書き方を示します。

・HTML3.2

各社ブラウザが独自に普及させたタグなどを含んだHTMLです。多くのブラウザがサポートはしていますが、レイアウトなどの見栄えを表現するための要素や属性を多く含んでいる一昔前のバージョンです。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

・HTML4.0 Strict

本来の1番理想的なHTML4.0を書きたい場合には、これを指定してください。ただし、レイアウトなどの見栄えを表現するための要素や属性(廃止予定のもの)は一切使うことができません。各社ブラウザが、HTML4.0とスタイルシートを完全にはサポートしていない現在、これに沿ったページを作る場合には様々な制限が出てくることでしょう。また、各要素を配置できる位置などについても、他と比較すると厳しい制限があります。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN"
 "http://www.w3.org/TR/REC-html40/strict.dtd">
```

・HTML4.0 Transitional

各社ブラウザの対応状況からすると、最も利用しやすいHTMLです。上記のHTML4.0 Strictをいきなり採用するには、現状では多少ムリがあるとの考えから作られたと思われる暫定的なバージョンです。廃止予定の要素や属性が使用可能で、各要素を配置できる位置についても自由度が高くなっています。ただし、フレームは使用できません。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
```

・HTML4.0 Frameset

上記のHTML4.0 Transitionalで、フレームを使用できるようにしたものです。その文書内に、フレームに関連する要素を含む場合に使用します。フレームに関連する部分を除くと、HTML4.0 Transitionalと全く同じものです。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
 "http://www.w3.org/TR/REC-html40/frameset.dtd">
```

HTML

HTMLに最低限必要な要素

<HTML>～</HTML>
<HEAD>～</HEAD>
<TITLE>～</TITLE>
<BODY>～</BODY>

解説

HTML文書では、まず先頭に<!DOCTYPE～>を書いて、それ以外の内容はすべて<HTML>～</HTML>の範囲内に書きます。<HTML>～</HTML>の中には、<HEAD>～</HEAD>と<BODY>～</BODY>を順に1つずつ入れてください。<HEAD>～</HEAD>の範囲にはその文書に関連する情報を、<BODY>～</BODY>の範囲にはその文書の実際の内容となるデータを入れることができます。また、<HEAD>～</HEAD>の範囲内には<TITLE>～</TITLE>を使用して、必ずその文書のタイトルを入れるようにしてください。ここで指定したタイトルは、一般的なブラウザではウインドウのタイトルバーに表示されるほか、「お気に入り」や「ブックマーク」として登録した場合のタイトルにもなります。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
        …この文書に関する情報…
<TITLE>この文書のタイトル</TITLE>
        …この文書に関する情報…
</HEAD>
<BODY>
        …この文書の実際の内容…
</BODY>
</HTML>
```

lang="ja" →「アクセシビリティ」の「何語で書かれているかを指定する」：P.242参照

背景色を指定する

HTML

<BODY bgcolor="色">～</BODY>

Internet Explorer	Netscape Navigator
背景色 このように、背景全体が指定した色で塗りつぶされます。しかし、スタイルシートを使うのがスマートなやり方です。	**背景色** このように、背景全体が指定した色で塗りつぶされます。しかし、スタイルシートを使うのがスマートなやり方です。

解説

ページ全体の背景を、指定した色で表示します。
ただし、この指定方法は、将来的に廃止されることになっています。背景色を指定する場合には、スタイルシートを利用するようにしてください。
また、一般的にブラウザは、印刷時に背景の色は印刷しないように設定されています。そのため、背景に黒などの濃い色を、文字に白などの薄い色を指定していると、結果として文字が印刷されないことになりますので注意してください。設定によって印刷することもできますが、その方法を知らない人は意外に多いようです。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>背景色を指定する</TITLE>
</HEAD>
<BODY bgcolor="#FFCC00">
<H1>背景色</H1>
<P>
このように、背景全体が指定した色で塗りつぶされます。
しかし、スタイルシートを使うのがスマートなやり方です。
</P>
</BODY>
</HTML>
```

「HTMLについて」の「色の指定方法」:P.25参照
「スタイルシート」の「背景色を指定する」:P.196参照
巻末付録「カラーチャート1～3」:巻末参照

HTML

背景画像を指定する

<BODY background="*画像のURL*">〜</BODY>

Internet Explorer

背景画像

このように、背景全体に指定した画像が配置されます。
しかし、スタイルシートを使うのがスマートなやり方です。

Netscape Navigator

背景画像

このように、背景全体に指定した画像が配置されます。
しかし、スタイルシートを使うのがスマートなやり方です。

背景画像として使用したファイル

解説

ページ全体の背景を、指定した画像をタイル状に並べた状態にします。
画像ファイルとしては、一般的なGIF形式のほか、JPEG形式やPNG形式も使用できます。回線の状況などによって画像がすぐにはロードされない場合もありますので、文字がハッキリと読めるような背景色も同時に指定しておくとよいでしょう。
ただし、この指定方法は、将来的に廃止されることになっています。背景画像を指定する場合には、スタイルシートを利用するようにしてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>背景画像を指定する</TITLE>
</HEAD>
<BODY bgcolor="#336666" background="back.gif" text=
"#FFFFFF">
<H1>背景画像</H1>
<P>
このように、背景全体に指定した画像が配置されます。
しかし、スタイルシートを使うのがスマートなやり方です。
</P>
</BODY>
</HTML>
```

「HTMLについて」の「ファイルの位置の指定方法 - 絶対URLと相対URL -」:P.25参照
「スタイルシート」の「背景画像を指定する」:P.197参照

NN2.0
NN3.0
NN4.0

IE3.0
IE4.0
IE5.0

HTML

基本となる文字色を設定する

```
<BODY text="色" link="色" vlink="色" alink="色">～</BODY>
```

text	リンクしていない普通の部分の文字色
link	リンクしている部分の文字色（リンク先をまだ見ていない場合、つまりリンク先がキャッシュされていない場合の文字色）
vlink	リンクしている部分の文字色（リンク先をすでに見た場合、つまりリンク先がキャッシュされている場合の文字色）
alink	リンクしている部分をクリックした時の文字色（リンク部分の上で、マウスボタンを押してから離すまでの文字色）

Internet Explorer

textで指定するのは、このような普通のテキストの色です。linkの解説は、まだ見ていないが、vlinkの解説はすでに見たという場合に、このような色になります。alinkの解説を見ようとしてマウスで押すと、その間だけ色が変化します。

Netscape Navigator

textで指定するのは、このような普通のテキストの色です。linkの解説は、まだ見ていないが、vlinkの解説はすでに見たという場合に、このような色になります。alinkの解説を見ようとしてマウスで押すと、その間だけ色が変化します。

解説

ページ全体を通しての文字色を設定します。
通常の文字と、リンクしている部分の3種類の状態の色をそれぞれ指定することができます。これらを指定する場合は、背景色の指定も含めてすべてセットで指定するようにしてください。そうでない場合は、ユーザーの設定によっては、文字が見にくくなってしまう可能性があります。
ただし、これらの指定方法は、将来的に廃止されることになっています。文字色を指定する場合には、スタイルシートを利用するようにしてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>文字色を設定する</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF" text="#000033" link="#0033FF"
 vlink="#666699" alink="#FF3300">
<P>
textで指定するのは、このような普通のテキストの色です。<A href="kaisetsu1.
html">linkの解説</A>は、まだ見ていないが、<A href="kaisetsu2.
html">vlinkの解説</A>はすでに見たという場合に、このような色になります。
<A href="kaisetsu3.html">alinkの解説</A>を見ようとしてマウスで
押すと、その間だけ色が変化します。
</P>
</BODY>
</HTML>
```

「HTMLについて」の「色の指定方法」:P.25参照
「スタイルとレイアウト」の「文字色を指定する」:P.67参照
「スタイルシート」の「文字色を指定する」:P.195参照
「スタイルシート」の「リンク部分のスタイルを指定する」:P.228参照
巻末付録「カラーチャート1～3」:巻末参照

HTML キーワードや内容の説明、制作者名などを入れる

```
<META name="author" content="制作者名">
<META name="description" content="内容の説明">
<META name="keywords" content="キーワード1, キーワード2, … ">
```

解説

そのページの制作者名、内容の説明、キーワードなどを指定します。
キーワードは、半角のカンマ(,)で区切って複数指定することができます。これらの情報は、画面上には表示されませんが、サーチエンジンが情報を収集する場合などに利用されます。特にキーワードとしてあたえる言葉は、そのページがうまく検索されるかどうかに関わるものですので、慎重に考えて指定してください。
これらの情報は、<HEAD>～</HEAD>の範囲に指定する必要があります。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>検索されやすいページの作り方</TITLE>
<META name="author" content="岡蔵　龍一">
<META name="description" content="キーワードによって、検索される
率を高める必殺テクニック集！">
<META name="keywords" content="検索,サーチエンジン,ヒット率,META,
キーワード,keyword,ロボット,サーチ">
</HEAD>
<BODY>
   ∫
</BODY>
</HTML>
```

HTML 文字セットやスタイルシート・スクリプトの言語を指定する

```
<META http-equiv="Content-Type" content="text/html;
 charset=文字セット">
<META http-equiv="Content-Script-Type" content="スクリプトの
タイプ">
<META http-equiv="Content-Style-Type" content="スタイルシー
トのタイプ">
```

文字セット	JIS：ISO-2022-JP, シフトJIS：Shift_JIS, EUC：EUC-JP
スクリプトのタイプ	JavaScript：text/javascript, VBScript：text/vbscript
スタイルシートのタイプ	Cascading Style Sheet：text/css

解説

1番目の書式は、そのHTML文書の文字セットを示します。
2番目の書式は、その文書で使用されているデフォルトのスクリプト言語を示します。イベントハンドラを使用する場合は、そこに記述されている言語を特定する手段がありませんので、必ず指定するようにしてください。
3番目の書式は、その文書で使用されているデフォルトのスタイルシート言語を示します。style属性を使用してスタイルを指定する場合は、そこに記述されている言語を特定する手段がありませんので、必ず指定するようにしてください。実際には、デフォルトの言語を指定しなくても、事実上のデフォルト言語(JavaScriptやCSS)で動作しますが、指定しておくのが正しい書き方です。
なお、これらの情報は、<HEAD>〜</HEAD>の範囲に指定する必要があります。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<META http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<META http-equiv="Content-Script-Type" content="text/
javascript">
<META http-equiv="Content-Style-Type" content="text/css">
<TITLE>META要素のサンプル</TITLE>
</HEAD>
<BODY style="color: #000033" onLoad="alert('読み込み完了！')">
  ∫
</BODY>
</HTML>
```

「スタイルシート」の「タグ内にスタイルシートを組み込むための属性」：P.192参照
「JavaScriptについて」の「イベントハンドラ」：P.260参照
注意「文字セットについて」：P.37参照

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML

自動的にページを読み込む

```
<META http-equiv="refresh" content="秒数">
<META http-equiv="refresh" content="秒数;URL=移動先URL">
```

Internet Explorer

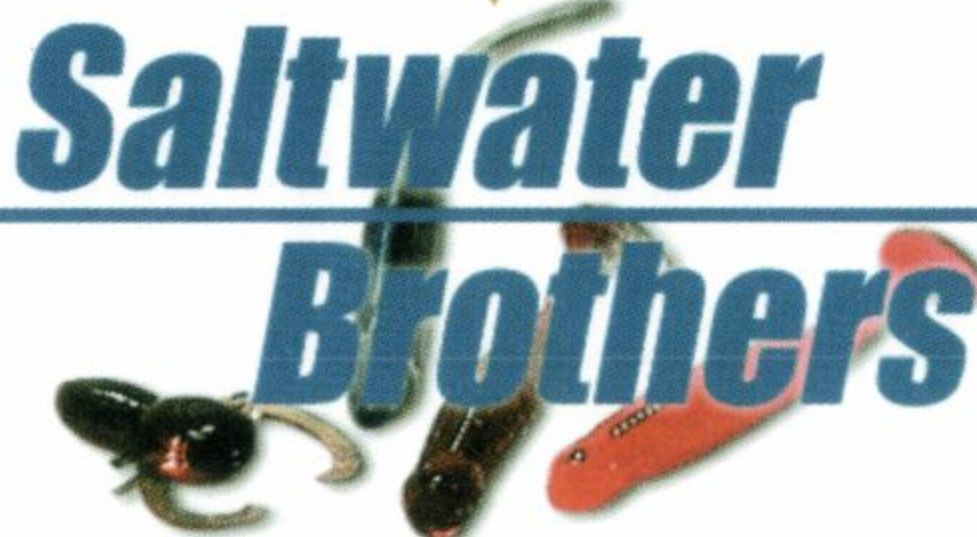
ソルブラは移動しました

ソルトウォーター・ブラザーズは以下のURLに移動しました。
新URL： http://www.elfish.com/~fish/

Netscape Navigator

ソルブラは移動しました

ソルトウォーター・ブラザーズは以下のURLに移動しました。
新URL： http://www.elfish.com/~fish/

Saltwater Brothers

解説

指定した秒数後に、自動的にページの読み込みを開始します。
移動先のURLを指定した場合はそのページを読み込みますが、指定していない場合は同じページを再読み込み(リロード)します。
この要素は、<HEAD>～</HEAD>の範囲に配置してください。
この機能は、ページのURLを変更した場合などに、元の古いページ(「引っ越しました」のページ)でよく利用されるものですが、すべてのブラウザでこの機能を利用できるわけではありません。自動的に別のページに移動させる場合には、そのページへのリンクもつけておくとよいでしょう。
なお、アクセシビリティを考慮するのであれば、この機能自体を利用しないようにしてください。

基本的な内容を指定する

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>移動しました</TITLE>
<META http-equiv="refresh" content="10;URL=http://www.
elfish.com/~fish/">
</HEAD>
<BODY>
<H1>ソルブラは移動しました</H1>
<P>
ソルトウォーター・ブラザーズは以下のURLに移動しました。<BR>
新URL:<A HREF="http://www.elfish.com/~fish/">
http://www.elfish.com/~fish/</A>
</P>
</BODY>
</HTML>
```

「アクセシビリティ」の「アクセシビリティについて」:P236参照

注意

文字セットについて

電子メールの場合はJISコードを使うことが常識となっていますが、HTML文書については、JIS・シフトJIS・EUC・UNICODEのいずれも正式なものとして利用することができます。前項の説明にもありますが、これらをMETA要素の「charset=」で指定する場合には、次のものを指定します。大文字と小文字の区別はされません。

JIS	ISO-2022-JP
シフトJIS	Shift_JIS
EUC	EUC-JP
UNICODE	UTF-8

また、シフトJISとEUCが正式なものとして登録される以前には、次の文字セットを利用できるブラウザ(Netscape Navigator 2.0)もありました。

シフトJIS	x-sjis
EUC	x-euc-jp

作成したHTML文書の文字セットとMETA要素で指定する文字セットが違っている場合には、文字化けしてしまいますので注意してください(現在、広く使われているパソコンでは、シフトJISが採用されています)。

ただし、本来は文字セットはMETA要素で指定するのではなく、サーバー側がMIMEヘッダの中のcharsetパラメータというものでブラウザに通知することになっています。ここでは詳しい解説は省略しますが、META要素で指定する方法は、現状を考慮した補助的な方法であることも知っておいてください。

HTML

NN4.0 IE3.0 IE4.0 IE5.0

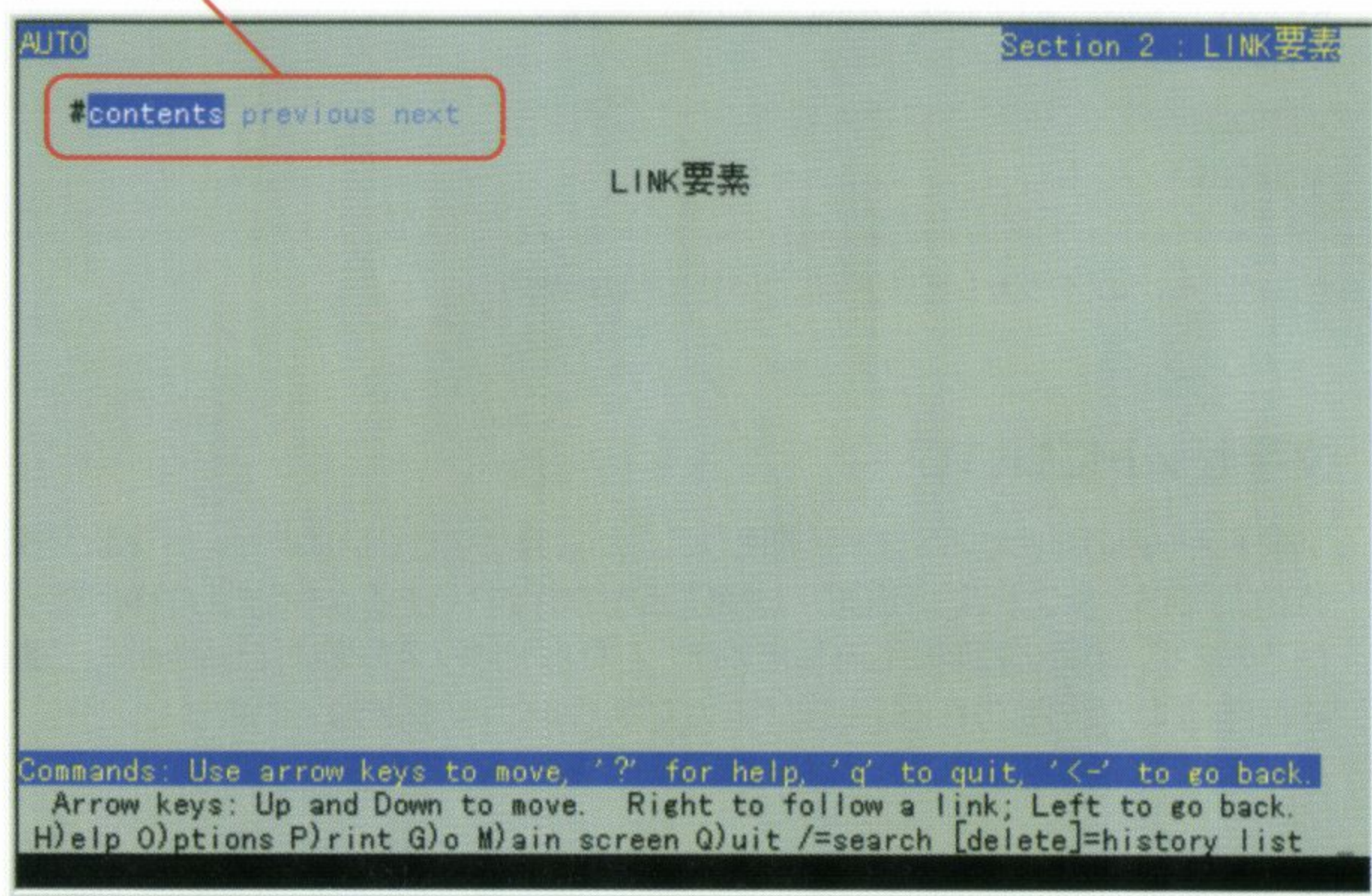
他のページとの関係を表す

```
<LINK rel="関係" href="URL">
<LINK rev="関係" href="URL">
```

rel	このページから見た、URLで示したページとの関係
rev	URLで示したページから見た、このページとの関係

Lynxでの表示例

```
AUTO                                              Section 2 : LINK要素
   #contents previous next
                                  LINK要素

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

解説

現在のページと、関連する他のページとの関係を示します。
例えば、前のページや次のページを示す場合などに使用されます。具体的な関係を示す値については、右ページのTip「ページ同士の関係を表す値」を参照してください。
現状では、多くのブラウザがこの情報を利用できるようにはなっていません。しかし、テキストブラウザのLynxでは、ナビゲーションシステムとして利用できますし、これらの情報はサーチエンジンを始めとする様々な用途に利用されることが考えられます。他のページとの関係を示すことは、決して意味のないことではありません。
なお、これらの情報は、<HEAD>～</HEAD>の範囲で指定する必要があります。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>Section 2 : LINK要素</TITLE>
<LINK rel="contents" href="mokuji.html">
<LINK rel="previous" href="section1.html">
<LINK rel="next" href="section3.html">
<LINK rel="stylesheet" href="default.css" type="text/css">
</HEAD>
<BODY>
<H1>LINK要素</H1>
  ∫
</BODY>
</HTML>
```

「スタイルシート」の「別ファイルのスタイルシートを読み込む」:P.190参照

Tip

ページ同士の関係を表す値

LINK要素とA要素では、rel属性とrev属性を使用して、他のページとの関係を表すことができます。その関係を示す値として、HTML4.0では以下のものが定義されています(大文字・小文字は区別されません)。

Alternate	別バージョン
Stylesheet	外部スタイルシート
Start	最初のページ
Next	次のページ
Prev	前のページ
Contents	目次
Index	索引
Glossary	用語集
Copyright	著作権に関するページ
Chapter	章
Section	節
Subsection	項
Appendix	付録
Help	ヘルプ
Bookmark	ブックマーク集

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML

パスの基準となるURLを指定する

```
<BASE href="URL">
<BASE href="URL" target="ターゲット名">
```

解説

そのページで使用する相対URLの基準となる絶対URLを設定します。
この指定を行うと、以降そのページで指定する相対URLは、すべてここで指定した絶対URLを基準としたものとして認識されます。この指定を行わなかった場合には、現在のページの位置が基準となります。target属性を指定すると、リンク先のページを開くデフォルトのフレームやウインドウを指定することができます。
なお、BASE要素は、<HEAD>〜</HEAD>の範囲で指定してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<BASE href="http://www.sample.com/~mariko/index.html"
 target="contentframe">
<TITLE>真理子のペンギン大好き!!:メニュー</TITLE>
</HEAD>
<BODY>
   ∫
<A href="diary/intro.html">日記のコーナー</A>
   ∫
</BODY>
</HTML>
```

「HTMLについて」の「ファイルの位置の指定 - 絶対URLと相対URL -」:P.25参照
Tip「ターゲット名について」:P.179参照

注意

target属性を指定する場合の注意

このサンプルのリンク部分の相対URL「diary/intro.html」は、以下に示す絶対URLとして認識されて「contentframe」という名前のフレームに表示されます。もし、このフレームがない場合には、新しいウインドウに表示されます。

```
http://www.sample.com/~mariko/diary/intro.html
```

HTML

内容に関する問い合わせ先を示す

<ADDRESS>～</ADDRESS>

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

Internet Explorer

*各製品に関するお問い合わせは、電子メールアドレス「order@abc.co.jp」または、フリーダイヤル「**0120-123-456**」まで。*

Netscape Navigator

*各製品に関するお問い合わせは、電子メールアドレス「order@abc.co.jp」または、フリーダイヤル「**0120-123-456**」まで。*

解説

そのページの内容に関する問い合わせ先や、連絡先が書かれている部分であることを示します。
具体的には、電子メールアドレス・制作者や担当者名・会社名・住所・電話番号・FAX番号などを示す場合に使用します。一般的なブラウザではイタリックで表示されますが、スタイルシートによって変更することも可能です。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>株式会社ABC:新製品一覧</TITLE>
</HEAD>
<BODY>
  ～
<ADDRESS>
各製品に関するお問い合わせは、電子メールアドレス「
<A href="mailto:order@abc.co.jp">order@abc.co.jp</A>」
または、フリーダイヤル「<B>0120-123-456</B>」まで。
</ADDRESS>
</BODY>
</HTML>
```

「スタイルシート」の「フォントスタイルを指定する」:P.210参照

HTML

コメントを入れる

<!--コメント文-->

解説

ページ内にコメントを入れておく場合に使用します。
この部分が実際に表示されることはありませんし、機能的にも一切影響をあたえませんので、更新時の注意書きなどを入れておくと便利です。
コメントには任意の文字を入れることができますが、ハイフンを連続して入れること(--など)は避けてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>社内ニュース</TITLE>
</HEAD>
<BODY>
<H1>社内ニュース</H1>
<P>
<!--更新時には、以下の行も忘れずに更新すること！-->
最終更新日:1999年9月1日(水)
</P>
<P>
さて、今月の～
</P>
</BODY>
</HTML>
```

HTML

見出しを表す

<H1>～</H1>
<H2>～</H2>
<H3>～</H3>
<H4>～</H4>
<H5>～</H5>
<H6>～</H6>

Internet Explorer

HTMLをマスターしよう
基本構造
HTML要素
HEAD要素
BODY要素
テキスト
P要素
H1～H6要素

Netscape Navigator

HTMLをマスターしよう
基本構造
HTML要素
HEAD要素
BODY要素
テキスト
P要素
H1～H6要素

解説

その部分が見出し(Heading)であることを示します。1～6の数字は見出しのレベルを表しており、<H1>が1番上のレベルの大見出し、<H6>が1番下のレベルの小見出しというように6段階まで用意されています。
一般的なブラウザでは、太字で上のレベルの見出しほど大きな文字で表示されます。
見出しとして画像を使う場合でも、このタグで囲うようにしてください。そうすれば、画像を表示できない状態の時でも、IMG要素のalt属性で指定した文字が見出しとして機能します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>見出しサンプル</TITLE>
</HEAD>
```

```
<BODY>
<H1>HTMLをマスターしよう</H1>
<H2>基本構造</H2>
<H3>HTML要素</H3>
<H3>HEAD要素</H3>
<H3>BODY要素</H3>
<H2>テキスト</H2>
<H3>P要素</H3>
<H3>H1～H6要素</H3>
</BODY>
</HTML>
```

「スタイルとレイアウト」の「行揃えを指定する」:P.70参照
「画像とマルチメディア」の「画像を配置する」:P.123参照
「スタイルシート」の「行揃えを指定する」:P.207参照

Tip Lynxでの見出しの表示例

見出しのレベルの違い(H1～H6)は、必ずしも大きさで表現されるわけではありません。例えば、テキストブラウザのLynxでは、上のサンプルを表示させると次のようになります。

見出しのレベルを大きさではなく、表示位置で示しているわけです。HTMLは表示方法を指定するものではありませんので、ブラウザによっては見出しをこのように表現したり、全く別の方法で表現するものもあるかもしれません。

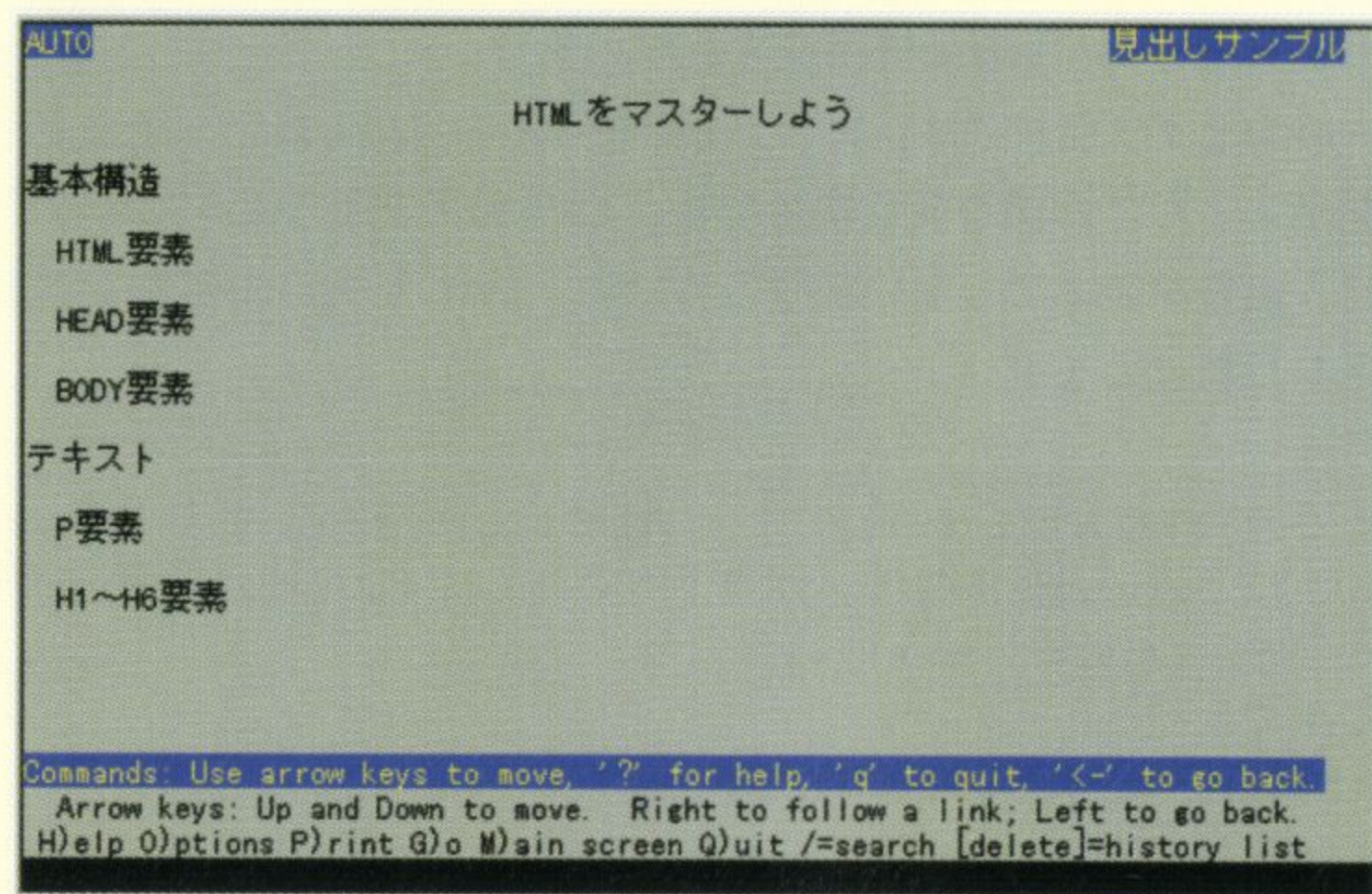
```
AUTO                                                    見出しサンプル
                    HTMLをマスターしよう
基本構造
  HTML要素
  HEAD要素
  BODY要素
テキスト
  P要素
  H1～H6要素

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

HTML

段落を表す

<P>〜</P>

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

Internet Explorer

読みやすさを考える

紙に書かれた文章とモニタに表示される文章では、どちらが読みやすいでしょうか？ほとんどの人は、紙の方が読みやすいと答えることでしょう。それには、様々な理由があるとは思うのですが、ここでは「どうすればモニタでも読みやすくすることができるか」を考えてたいと思います。

まず、画面上に隙間なく続く長い文章は、見ただけで読む気がしなくなるものです。環境によっては改行幅が極端に狭く、次の行の先頭が分かりにくくて読んでいられない場合もあります。そのような意味では、段落の間を1行あける(空間を作る)P要素の表現方法は適切だと考えられます。

Netscape Navigator

読みやすさを考える

紙に書かれた文章とモニタに表示される文章では、どちらが読みやすいでしょうか？ほとんどの人は、紙の方が読みやすいと答えることでしょう。それには、様々な理由があるとは思うのですが、ここでは「どうすればモニタでも読みやすくすることができるか」を考えてみたいと思います。

まず、画面上に隙間なく続く長い文章は、見ただけで読む気がしなくなるものです。環境によっては改行幅が極端に狭く、次の行の先頭が分かりにくくて読んでいられない場合もあります。そのような意味では、段落の間を1行あける(空間を作る)P要素の表現方法は適切だと考えられます。

解説

その部分が1つの段落(Paragraph)であることを示します。日本語の場合、始めに1文字分空けることによって段落を示す場合が多いのですが、一般的なブラウザでは、この要素の前で改行して、更に1行空けた状態で表示されます。
終了タグ</P>は、次に新しい段落が開始されたり、見出しなどが配置されることによってそれが終了したことが分かるため、省略することが可能です。しかし、その範囲を明確に示すためにも、終了タグは常に付けるようにした方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>段落サンプル</TITLE>
</HEAD>
<BODY>
<H1>読みやすさを考える</H1>
<P>
紙に書かれた文章とモニタに表示される文章では、(中略) 考えてみたいと思います。
</P>
<P>
まず、画面上に隙間なく続く長い文章は、(中略)P要素の表現方法は適切だと考えられます。
</P>
</BODY>
</HTML>
```

「スタイルとレイアウト」の「行揃えを指定する」: P.70参照
「スタイルシート」の「行揃えを指定する」: P.207参照

HTML

引用された短い文章を表す

<Q>〜</Q>
<Q cite="*URL*">〜</Q>
<Q lang="*言語コード*">〜</Q>

cite	引用先のURL
lang	何語の引用符を使用するか

Internet Explorer

これらの要素がどのような役割をするものかを知っていれば インデントをする目的で引用を表す要素を使用しないという も理解できるはずだ。

Netscape Navigator

これらの要素がどのような役割をするものかを知っていれば、インデントをする目的で引用を表す要素を使用しないということも理解できるはずだ。

Lynx

AUTO　短い引用サンプル

これらの要素がどのような役割をするものかを知っていれば、" インデントをする目的で引用を表す要素を使用しない" ということも理解できるはずだ。

解説

その部分が、インラインで使用する短い引用文であることを示します。
cite属性を使用して、引用先のURLを示すこともできます。
この要素がサポートされているブラウザでは、指定した範囲の前後に自動的に引用符が付けられますので、自分では引用符を付けないようにしてください。引用符は言語によって異なります。日本語の引用符であることを示す場合には、「lang="ja"」と指定してください。英語の場合には、引用符として「"」が使用されます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>短い引用サンプル</TITLE>
</HEAD>
<BODY>
<P>
これらの要素がどのような役割をするものかを知っていれば、<Q cite="http://
www.w3.org/TR/WAI-WEBCONTENT/" lang="ja">インデントをする目的で
引用を表す要素を使用しない</Q>ということも理解できるはずだ。
</P>
</BODY>
</HTML>
```

引用された長い文章を表す

<BLOCKQUOTE>～</BLOCKQUOTE>
<BLOCKQUOTE cite="*URL*">～</BLOCKQUOTE>

cite	引用先のURL

Internet Explorer

例えば、W3Cのアクセシビリティ・ガイドライン1.0 の中には、次のような一節があります。

> 仕様に従った正しいタグ付けを行わないことは、アクセシビリティ上の妨げとなります。表示上の効果を目的として間違ったタグ付けを行うことは、一般的ではないソフトウェアを利用しているユーザーにとって、ページの構成を理解することやサイト内を見てまわることを難しくします。

Netscape Navigator

例えば、W3Cのアクセシビリティ・ガイドライン1.0 の中には、次のような一節があります。

> 仕様に従った正しいタグ付けを行わないことは、アクセシビリティ上の妨げとなります。表示上の効果を目的として間違ったタグ付けを行うことは、一般的ではないソフトウェアを利用しているユーザーにとって、ページの構成を理解することやサイト内を見てまわることを難しくします。

解説

その部分が、ブロックで使用する長い引用文であることを示します。
cite属性を使用して、引用先のURLを示すこともできます。
一般的なブラウザでは、上下に1行分のスペースがとられ、左右がインデントされた状態で表示されます。このため、かつては左右のマージンをとる目的で利用されることも多かったのですが、現在ではスタイルシートで自由にマージンを設定することができますので、そちらを利用するようにしましょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>長い引用サンプル</TITLE>
</HEAD>
<BODY>
```

```
<P>
例えば、W3Cのアクセシビリティ・ガイドライン1.0 の中には、次のような一節があります。
</P>

<BLOCKQUOTE cite="http://www.w3.org/TR/WAI-WEBCONTENT/">
<P>
仕様に従った正しいタグ付けを行わないことは、アクセシビリティ上の妨げとなります。表示上の効果を目的として間違ったタグ付けを行うことは、一般的ではないソフトウェアを利用しているユーザーにとって、ページの構成を理解することやサイト内を見てまわることを難しくします。
</P>
</BLOCKQUOTE>

</BODY>
</HTML>
```

「スタイルシート」の「マージンを設定する」:P.205参照

コラム

引用文に対するブラウザの対応状況

この原稿の執筆時点(1999年8月)では、BLOCKQUOTE要素がブロックとして表示され、左右がインデントされて表示されることを除けば、ほとんどのブラウザが引用を表す要素には対応していないと言っていいでしょう。W3Cの提供するAmayaやLynxなどのブラウザでは、Q要素の両端を「"」で囲いますが、それ以外のブラウザでは何も起こりません。cite属性で指定した引用先のURLを利用できるのは、現在ではAmayaだけのようです。

引用(参照)先のタイトルや名前を表す

<CITE>〜</CITE>

Internet Explorer

詳細は、*[ISO-9999]*を参照してください。

Netscape Navigator

詳細は、*[ISO-9999]*を参照してください。

解説

引用する新聞や書籍などのタイトルや名前(出典)を表す場合や、規格などの参照先を示す場合に使用します。一般的なブラウザでは、イタリックで表示されます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>引用（参照）先サンプル</TITLE>
</HEAD>
<BODY>
<P>
詳細は、<CITE>[ISO-9999]</CITE> を参照してください。
</P>
</BODY>
</HTML>
```

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML

強調したい部分を表す

<EM>～</EM>
<STRONG>～</STRONG>

Internet Explorer

【注意】*この原稿の執筆時点*では、必ずしも最新版のブラウザが、この本で紹介している内容をすべてサポートしているわけではありません。

Netscape Navigator

【注意】*この原稿の執筆時点*では、必ずしも最新版のブラウザが、この本で紹介している内容をすべてサポートしているわけではありません。

解説

その部分が、強調されていることを示します。
EM要素は普通の強調を、STRONG要素はより強い強調であることを示します。一般的なブラウザでは、EM要素はイタリックで、STRONG要素は太字で表示されます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>強調サンプル</TITLE>
</HEAD>
<BODY>
<P>
<STRONG>【注意】</STRONG>
<EM>この原稿の執筆時点</EM>では、必ずしも最新版のブラウザが、この本で紹介して
いる内容をすべてサポートしているわけではありません。
</P>
</BODY>
</HTML>
```

HTML

コンピュータ関連のテキストを表す

<KBD>～</KBD>　←入力文字
<SAMP>～</SAMP>　←出力サンプル
<CODE>～</CODE>　←ソースコード
<VAR>～</VAR>　←変数・引数

Internet Explorer

まずは、コマンドラインからDIRと入力してみましょう。もし、間違って入力した場合には、次のようなメッセージが表示されます。

コマンドまたはファイル名が違います。

...

次に示す例のように、通常カウンタとなる変数には、*i*が使用されます。

```
for(var i in document)
    document.write(i + "<BR>")
```

Netscape Navigator

まずは、コマンドラインからDIRと入力して みましょう。もし、間違って入力した場合には、次の ようなメッセージが表示されます。

コマンドまたはファイル名が違います。

...

次に示す例のように、通常カウンタとなる変数には、*i*が使用されます。

```
for(var i in document)
    document.write(i + "<BR>")
```

解説

KBD要素は、その部分がキーボードから入力する文字であることを示します。
SAMP要素は、プログラムなどによって出力される内容のサンプルを示す場合に使用します。
CODE要素は、プログラムなどのソースコードを示す場合に使用します。ソースコード中の空白による字下げが大きな意味を持つ場合には、同時にPRE要素も使用すると便利です。
VAR要素は、変数や引数を示す場合に使用します。
一般的なブラウザでは、KBD要素・SAMP要素・CODE要素は等幅フォントで、VAR要素はイタリックで表示されます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>コンピュータ関連のテキストサンプル</TITLE>
</HEAD>
<BODY>

<P>
まずは、コマンドラインから<KBD>DIR</KBD>と入力して
みましょう。もし、間違って入力した場合には、次の
ようなメッセージが表示されます。
</P>

<P>
<SAMP>コマンドまたはファイル名が違います。</SAMP>
</P>

<P>・・・</P>

<P>
次に示す例のように、通常カウンタとなる変数には、
<VAR>i</VAR> が使用されます。
</P>

<PRE><CODE>
for(var i in document)
    document.write(i + "&lt;BR&gt;")
</CODE></PRE>
</BODY>
</HTML>
```

「スタイルとレイアウト」の「空白や改行を入力した通りに表示させる」:P.69参照

定義語を表す

<DFN>～</DFN>

Internet Explorer

ウェブ・コンテンツにアクセスして、それを利用するソフトウェアを広く*ユーザーエージェント*と呼んでいます。

Netscape Navigator

ウェブ・コンテンツにアクセスして、それを利用するソフトウェアを広くユーザーエージェントと呼んでいます。

解説

その部分が、定義語(専門的な用語)であることを示します。
一般的なブラウザでは、イタリックで表示するものもありますが、特にスタイルが変化しないものもあります。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>定義語サンプル</TITLE>
</HEAD>
<BODY>
<P>
ウェブ・コンテンツにアクセスして、それを利用するソフトウェアを広く<DFN>ユーザー
エージェント</DFN>と呼んでいます。
<P>
</BODY>
</HTML>
```

略語・頭字語を表す

```
<ABBR title="文字列">～</ABBR>                  ←略語
<ACRONYM title="文字列">～</ACRONYM>            ←頭字語
```

IE4.0
IE5.0

title	省略していない状態の言葉(文字列)

Internet Explorer

SOHOの方であれば、TCP/IPでLANを組んでいる人も多いでしょう。

SOHOの方であれば、TCP/IPでLANを組んでいる人も多いでしょう。
Small Office Home Office

Netscape Navigator

SOHOの方であれば、TCP/IPでLANを組んでいる人も多いでしょう。

解説

その部分が、略語であることを示します。
その略語を1文字ずつ発音するもの(例：URL・HTML・CSS・FBI…)にはABBR要素を、1つの単語として発音するもの(例：SCSI・ROM・SOHO・NATO…)にはACRONYM要素を使用します。
どちらともつかない場合には、ABBR要素を使用するようにしてください。
title属性には、省略しない状態の言葉を指定します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>略語・頭字語サンプル</TITLE>
</HEAD>
<BODY>
<P>
<ACRONYM title="Small Office Home Office">SOHO</ACRONYM>
の方であれば、<ABBR title="Transmission Control Protocol/Inter
net Protocol">TCP/IP</ABBR>で<ACRONYM title="Local Area Ne
twork">LAN</ACRONYM>を組んでいる人も多いでしょう。
</P>
</BODY>
</HTML>
```

HTML

後から追加した部分を示す

<INS cite="*URL*" datetime="*追加日時*">〜</INS>

IE4.0 IE5.0

cite	追加した理由が書かれている文書のURL
datetime	追加した日時(ISO8601形式)

Internet Explorer

CENTER要素は、要素内容を横方向の中心に配置させます。
注意)この要素は、HTML4.0では廃止予定となりました。

Netscape Navigator

CENTER要素は、要素内容を横方向の中心に配置させます。
注意)この要素は、HTML4.0では廃止予定となりました。

解説

その部分が、後から追加した部分であることを示します。
W3Cの仕様のように、バージョン管理がされている文書に利用すると便利です。ブラウザによって表示方法は異なりますが、一般的には下線の付いた状態やイタリックなどで表示されます。また、ブラウザによっては、title属性で指定した短い説明をツールチップとして表示させることもできます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>追加部分サンプル</TITLE>
</HEAD>
<BODY>
<P>
CENTER要素は、要素内容を横方向の中心に配置させます。<INS cite="http://
www.w3.org/TR/REC-html40/appendix/changes.html" datetime
="1998-04-19T22:14:00+09:00">注意)この要素は、HTML4.0では廃止予定
となりました。
</INS>
</P>
</BODY>
</HTML>
```

後から削除した部分を示す

<DEL cite="*URL*" datetime="*削除日時*">～</DEL>

IE4.0
IE5.0

cite	削除した理由が書かれている文書のURL
datetime	削除した日時(ISO8601形式)

Internet Explorer

~~CENTERタグを利用すると、なんでも簡単にセンタリングできるので大変便利です。~~

~~CENTERタグを利用すると、なんでも簡単にセンタリングできるので大変便利です。~~
スタイルシートでするのが正しいそうです

Netscape Navigator

CENTERタグを利用すると、なんでも簡単にセンタリングできるので大変便利です。

解説

その部分が、後から削除された部分であることを示します。
W3Cの仕様のように、バージョン管理がされている文書に利用すると便利です。ブラウザによって表示方法は異なりますが、一般的には字消線が付けられた状態で表示されます。また、ブラウザによっては、title属性で指定した短い説明をツールチップとして表示させることもできます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>削除部分サンプル</TITLE>
</HEAD>
<BODY>
```

```
<P>
<DEL cite="http://www.w3.org/TR/REC-html40/appendix/changes.html" datetime="1997-12-31T21:30:00+09:00" title="スタイルシートでするのが正しいそうです">CENTERタグを利用すると、なんでも簡単にセンタリングできるので大変便利です。
</DEL>
</P>
</BODY>
</HTML>
```

更新日時の表し方

INS要素とDEL要素で指定する日時のフォーマットは、ISO8601に準拠した形式で、見た感じはちょっと難しそうです。しかし、基本的には年・月・日・時・分・秒とタイムゾーンを示すだけですので、分かってしまえば決して難しいものではありません。

以下に示す書式で、必要な部分を書き換えればOKです。年は4桁、その他は2桁で固定されていますので、注意してください。

HTML

ルビをふる

<RUBY>～</RUBY>　←ルビの対象範囲
<RT>～　←ルビ（ふりがな）
<RP>～　←ルビ未対応用の区切り記号

IE5.0

Internet Explorer

長万部(おしゃまんべ)とは、北海道の地名である。

Netscape Navigator

長万部（おしゃまんべ）とは、北海道の地名である。

解説

指定した範囲にルビ(ふりがな)をふります。
RUBY要素でルビをふる範囲を指定して、RT(Ruby Text)要素でルビとして小さく表示される文字を示します。RP要素は、これらの要素に対応していないブラウザが、ルビを普通の大きさの文字で続けて表示してしまわないように、ルビを囲う文字(通常は括弧)を指定します。
現在、一部のブラウザで利用可能ですが、これらの要素はW3CがHTMLの次期バージョンに含めるために検討中の仕様案であって、正式なものではありません。したがって、今後、その構造や要素の名前自体が変更される可能性もあります。

Sample

```
<HTML lang="ja">
<HEAD>
<TITLE>ルビサンプル</TITLE>
</HEAD>
<BODY>
<P>
<RUBY>長万部<RP>(<RT>おしゃまんべ<RP>)</RUBY>とは、北海道の地名である。
</P>
</BODY>
</HTML>
```

注意

RT要素・RP要素の終了タグ

W3Cの仕様案では、ルビをふる対象の言葉を示すRB要素があります。
また、ここではRT要素とRP要素の終了タグを省略した形で紹介しています。現在サポートしているブラウザでは、終了タグをつけると余計な空白が入れられるようです。

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML

特別な文字を表示させる

```
&lt;      ←<
&gt;      ←>
&quot;    ←"
&amp;     ←&
```

Internet Explorer

```
<HTML lang="ja">
<HEAD>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Netscape Navigator

```
<HTML lang="ja">
<HEAD>
</HEAD>
<BODY>
</BODY>
</HTML>
```

解説

HTMLでは、「<」と「>」はタグを表すために使用されますので、そのまま書くとそこがタグの一部だと解釈されてしまいます。
そのような特別な意味を持った文字を表示させたい場合には、「&○○;」という形式を使用します。これらは、必ず小文字で書くようにしてください。
実際には、この他にも様々な文字を表示させることが可能ですが、多くは日本語のフォントでは文字化けしてしまいます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>特別な文字サンプル</TITLE>
</HEAD>
<BODY>
<CODE>
&lt;HTML lang=&quot;ja&quot;&gt;<BR>
&lt;HEAD&gt;<BR>
&lt;/HEAD&gt;<BR>
&lt;BODY&gt;<BR>
&lt;/BODY&gt;<BR>
&lt;/HTML&gt;
</CODE>
</BODY>
</HTML>
```

付録「HTML4.0で定義されている特別な文字」:P.246参照

NN2.0
NN3.0
NN4.0

IE3.0
IE4.0
IE5.0

HTML

表示に関する指定をする

HTML

フォントスタイルを指定する

<B>～</B>	←太字
<I>～</I>	←イタリック
<TT>～</TT>	←等幅フォント
[～]	←上付き文字
_～	←下付き文字
<U>～</U>	←下線
<S>～</S>	←字消線
<STRIKE>～</STRIKE>	←字消線

Internet Explorer

これは**太字(bold)**
これは*イタリック(italic)*
これは等幅フォント(monospaced)
これは上付き文字(superscript)
これは下付き文字(subscript)
これは下線(underlined)
これは~~字消線(strike-through)~~
これは~~字消線(strike-through)~~

Netscape Navigator

これは**太字(bold)**
これは*イタリック(italic)*
これは等幅フォント(monospaced)
これは上付き文字(superscript)
これは下付き文字(subscript)
これは下線(underlined)
これは~~字消線(strike-through)~~
これは~~字消線(strike-through)~~

解説

指定した範囲のフォントスタイルを指定します。
これらは、文書の構造的な意味を持ってはいませんが、HTMLでは該当する適切な要素がない場合や、ブラウザが使いたい要素をサポートしていない場合に、自分なりの構造的な意味を持たせて補助的に使用すると便利です(特に太字やイタリック)。
HTML4.0では、古いバージョンとの互換の意味で、「字消線」を表す、まったく同じ意味の要素が2つあります(SとSTRIKE)。これらの要素のうち、特にU・S・STRIKEの各要素は、将来的に廃止されることになっていますので、代わりにスタイルシートを利用するとよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フォントスタイルサンプル</TITLE>
</HEAD>
<BODY>
```

```
<P>
これは<B>太字(bold)</B><BR>
これは<I>イタリック(italic)</I><BR>
これは<TT>等幅フォント(monospaced)</TT><BR>
これは<SUP>上付き文字(superscript)</SUP><BR>
これは<SUB>下付き文字(subscript)</SUB><BR>
これは<U>下線(underlined)</U><BR>
これは<S>字消線(strike-through)</S><BR>
これは<STRIKE>字消線(strike-through)</STRIKE>
</P>
</BODY>
</HTML>
```


「スタイルとレイアウト」の「フォントの種類を指定する」:P.66参照

「スタイルシート」の「フォントのスタイルを指定する」:P.210参照

「スタイルシート」の「フォントの太さを指定する」:P.212参照

「スタイルシート」の「フォントの種類を指定する」:P.215参照

Tip

リンク部分の下線を消したい

リンク部分の下線を消したい場合など、スタイルを指定したい場合とは逆に、スタイルをなくしたい場合もあると思います。そのような場合には、スタイルシートを利用します。

詳しい内容については、「スタイルシート」の「リンク部分のスタイルを指定する」(P.228)を参照していただくとして、次のようにスタイルを指定することで、リンク部分の下線を消すことができます。

```
A:link, A:visited, A:active { text-decoration: none }
```

HTML

フォントサイズを指定する

```
<FONT size="サイズ">～</FONT>
```

size	1～7（1が最小、7が最大。標準は3）

Internet Explorer

フォントの標準サイズは、3(このサイズ)です。
フォントサイズ1
フォントサイズ2
フォントサイズ3
フォントサイズ4
フォントサイズ5
フォントサイズ6
フォントサイズ7

Netscape Navigator

フォントの標準サイズは、3(このサイズ)です。
フォントサイズ1
フォントサイズ2
フォントサイズ3
フォントサイズ4
フォントサイズ5
フォントサイズ6
フォントサイズ7

解説

指定した範囲のフォントサイズを指定します。
サイズとしては1～7の数字を指定できますが、具体的な大きさは決まっていませんので、実際に表示される大きさはブラウザによって異なります。
この要素は将来的に廃止される予定となっていますので、フォントのサイズを指定する場合には、スタイルシートを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フォントサイズサンプル</TITLE>
</HEAD>
<BODY>
<P>
フォントの標準サイズは、3 (このサイズ) です。<BR>
<FONT size="1">フォントサイズ1</FONT><BR>
<FONT size="2">フォントサイズ2</FONT><BR>
<FONT size="3">フォントサイズ3</FONT><BR>
<FONT size="4">フォントサイズ4</FONT><BR>
<FONT size="5">フォントサイズ5</FONT><BR>
<FONT size="6">フォントサイズ6</FONT><BR>
<FONT size="7">フォントサイズ7</FONT>
</P>
</BODY>
</HTML>
```

「スタイルシート」の「フォントサイズを指定する」：P.213参照

HTML

フォントの基本サイズを指定する

<BASEFONT size="*サイズ*">

size	1～7(1が最小、7が最大。標準は3)

Internet Explorer

まおちゃんとおともだち

まおちゃんは、げんきなおんなのこです。
あるあさ、おうちをでると、おおきないぬ
あるいてきました。そのいぬは、しろくて
とてもやさしいかおをしています。

Netscape Navigator

まおちゃんとおともだち

まおちゃんは、げんきなおんなのこです。
あるあさ、おうちをでると、おおきないぬが
あるいてきました。そのいぬは、しろくて、
とてもやさしいかおをしています。

解説

この指定以降のフォントの基本サイズを設定します。ただし、見出しの大きさには影響しません。
サイズとしては1～7の数字を指定できますが、具体的な大きさは決まっていませんので、実際に表示される大きさはブラウザによって異なります。
この要素は将来的に廃止される予定となっていますので、フォントのサイズに関する指定をする場合には、スタイルシートを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フォントの基本サイズサンプル</TITLE>
</HEAD>
<BODY>
<BASEFONT size="5">
<H1>まおちゃんとおともだち</H1>
<P>
まおちゃんは、げんきなおんなのこです。あるあさ、おうちをでると、おおきないぬがある
いてきました。そのいぬは、しろくて、とてもやさしいかおをしています。
</P>
</BODY>
</HTML>
```

「スタイルシート」の「フォントサイズを指定する」:P.213参照

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML フォントを大きくする／小さくする

`<BIG>～</BIG>`	←大きく
`<SMALL>～</SMALL>`	←小さく
`<FONT size="+n">～</FONT>`	←n段階大きく
`<FONT size="-n">～</FONT>`	←n段階小さく

n	現在のサイズを基準として±の結果が1～7の範囲となる数字

Internet Explorer

これはSMALL要素で小さくしています。
これは標準サイズの文字です。
これはBIG要素で大きくしています。

フォントサイズ-2
フォントサイズ-1
標準サイズ（フォントサイズ3）
フォントサイズ+1
フォントサイズ+2
フォントサイズ+3
フォントサイズ+4

Netscape Navigator

これはSMALL要素で小さくしています。
これは標準サイズの文字です。
これはBIG要素で大きくしています。

フォントサイズ-2
フォントサイズ-1
標準サイズ（フォントサイズ3）
フォントサイズ+1
フォントサイズ+2
フォントサイズ+3
フォントサイズ+4

解説

BIG要素は標準のフォントサイズよりも大きく、SMALL要素は標準のフォントサイズよりも小さく表示します。
サイズを細かく指定することはできません。FONT要素を使用すると、現在のサイズに対して何段階大きく、または小さくするかを指定することができます。size属性に±をつけた数字で指定してください。ただし、フォントサイズは全部で7段階になっていますので、±した結果が1～7の範囲の数字になるようにしてください。
FONT要素は将来的に廃止される予定となっています。フォントのサイズに関する指定をする場合には、スタイルシートを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フォントを大きく／小さくサンプル</TITLE>
</HEAD>
<BODY>

<P>
<SMALL>これはSMALL要素で小さくしています。</SMALL><BR>
これは標準サイズの文字です。<BR>
<BIG>これはBIG要素で大きくしています。</BIG>
</P>

<P>
<FONT size="-2">フォントサイズ-2</FONT><BR>
<FONT size="-1">フォントサイズ-1</FONT><BR>
標準サイズ(フォントサイズ3)<BR>
<FONT size="+1">フォントサイズ+1</FONT><BR>
<FONT size="+2">フォントサイズ+2</FONT><BR>
<FONT size="+3">フォントサイズ+3</FONT><BR>
<FONT size="+4">フォントサイズ+4</FONT>
</P>

</BODY>
</HTML>
```

「スタイルシート」の「フォントサイズを指定する」:P.213参照

HTML

フォントの種類を指定する

<FONT face="フォント名,フォント名,・・・">～</FONT>

Internet Explorer(Win)

ここはゴシック体で、ここは
明朝体にしてみます。

Netscape Navigator(Win)

ここはゴシック体で、ここは
明朝体にしてみます。

Internet Explorer(Mac)

ここはゴシック体で、ここは
明朝体にしてみます。

Netscape Navigator(Mac)

ここはゴシック体で、ここは
明朝体にしてみます。

解説

指定した範囲のテキストを表示するフォントの種類を指定します。
フォント名は1つでも指定できますが、カンマで区切って複数指定することができます。その場合は、より先(左)に指定されているフォントで、ユーザーの環境で表示可能なものが採用されます。当然のことですが、フォント名は間違っていると(全角と半角の違いなど)正しく表示されませんので注意してください。
ただし、この要素は将来的に廃止される予定となっています。フォントの種類を指定する場合には、スタイルシートを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フォントの種類サンプル</TITLE>
</HEAD>
<BODY>
<P>
<FONT face="MS Pゴシック,中ゴシックBBB,Osaka" size="6">ここは
ゴシック体で、</FONT><FONT face="MS P明朝,リュウミンライトーKL,細明
朝体" size="6">ここは明朝体にしてみます。</FONT>
</P>
</BODY>
</HTML>
```

「スタイルシート」の「フォントの種類を指定する」:P.215参照
巻末付録「フォント表示見本」:巻末参照

HTML

文字色を指定する

<FONT color="色">～</FONT>

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

Internet Explorer

■ 文字色

FONT要素でテキストの色を変更できます。しかし、そのような目的にはスタイルシートを利用した方がよいでしょう。

Netscape Navigator

■ 文字色

FONT要素でテキストの色を変更できます。しかし、そのような目的にはスタイルシートを利用した方がよいでしょう。

解説

指定した範囲のテキストの色を指定します。
ページ全体を通しての文字色やリンク部分に関する色は、BODY要素の属性で指定することができます。
しかし、属性によって文字色を指定する方法は、将来的に廃止される予定となっていますので、かわりにスタイルシートを利用するようにしてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フォントカラー</TITLE>
</HEAD>
<BODY>
<H1>
<FONT color="#99CC00">■</FONT>
文<FONT color="#666666">字</FONT>色
</H1>
<P>
FONT要素でテキストの色を変更できます。しかし、そのような目的には<FONT color
="#FF0033">スタイルシート</FONT>を利用した方がよいでしょう。
</P>
</BODY>
</HTML>
```

「HTMLについて」の「色の指定方法」：P.25参照
「ページの基礎となる内容」の「基本となる文字色を設定する」：P.32参照
「スタイルシート」の「文字色を指定する」：P.195参照
巻末付録「カラーチャート1～3」：巻末参照

NN2.0
NN3.0
NN4.0

IE3.0
IE4.0
IE5.0

HTML

改行させる

Internet Explorer

HTMLを書くことは、簡単なようで難しい。
HTMLの正式な仕様を覚えることは、プログラミング言語を
覚えるよりも遥かに簡単なことだ。しかし、いざ実際に書き
始めてみると、仕様と実装のあまりの違いに・・・

Netscape Navigator

HTMLを書くことは、簡単なようで難しい。
HTMLの正式な仕様を覚えることは、プログラミング言語を
覚えるよりも遥かに簡単なことだ。しかし、いざ実際に書き
始めてみると、仕様と実装のあまりの違いに・・・

解説

この要素を入れると、テキストがそこで改行されます。
HTMLのソースの中で改行してもブラウザで表示する場合には反映されませんので、改行して表示させたい場合にはこの要素を使用します。
ただし、ユーザーが必ずしも制作者と同じサイズのフォントで見ているとは限りませんので、整形やレイアウトを目的として改行する場合には注意が必要です。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>改行サンプル</TITLE>
</HEAD>
<BODY>
<P>
HTMLを書くことは、簡単なようで難しい。<BR>
HTMLの正式な仕様を覚えることは、プログラミング言語を覚えるよりも遥かに簡単な
ことだ。しかし、いざ実際に書き始めてみると、仕様と実装のあまりの違いに・・・
</P>
</BODY>
</HTML>
```

表示に関する指定をする

空白や改行を入力した通りに表示させる

<PRE>～</PRE>

Internet Explorer

```
function resetRadio() {
    for(var i = 0; i < document.form1.type.length; i++) {
        if(document.form1.type[i].defaultChecked == true)
            document.form1.type[i].checked = true
        else
            document.form1.type[i].checked = fals
    }
}
```

Netscape Navigator

```
function resetRadio() {
    for(var i = 0; i < document.form1.type.length; i++) {
        if(document.form1.type[i].defaultChecked == true)
            document.form1.type[i].checked = true
        else
            document.form1.type[i].checked = false
    }
}
```

解説

指定した範囲のテキストを、入力した通りに等幅フォントで表示します。具体的には、スペースと改行が入力したままの状態で表示され、1行の長さがウインドウの幅より長くなっても、自動的に改行されなくなります。主にソースコードなどを表示させたい場合に使用しますが、タブは使わないようにしてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>空白や改行を入力した通りに表示させるサンプル</TITLE>
</HEAD>
<BODY>
<PRE><CODE>
function resetRadio() {
    for(var i = 0; i < document.form1.type.length; i++) {
        if(document.form1.type[i].defaultChecked == true)
            document.form1.type[i].checked = true
        else
            document.form1.type[i].checked = false
    }
}
</CODE></PRE>
</BODY>
</HTML>
```

「テキスト」の「コンピュータ関連のテキストを表す」:P.51参照

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML

行揃えを指定する

```
<P align="行揃え位置">～</P>
<Hn align="行揃え位置">～</Hn>
<DIV align="行揃え位置">～</DIV>
```

行揃え位置	left・right・center

※Hnは「見出しを表す要素」のH1～H6を表しています。

Internet Explorer

夜明けの行揃え

実をいうと、行揃えには「left, right, center」のほかに「justify」というものがある。いわゆる「両端揃え」のことである。しかし、色々試してはみたのだが、これが実際には・・・

【文:岡蔵　龍一】

- ご感想をおよせください -

(C) Copyright 1999 OKARYU,Inc. All Rights Reserved.

Netscape Navigator

夜明けの行揃え

実をいうと、行揃えには「left, right, center」のほかに「justify」というものがある。いわゆる「両端揃え」のこと である。しかし、色々試してはみたのだが、これが実際には・・・

【文:岡蔵　龍一】

- ご感想をおよせください -

(C) Copyright 1999 OKARYU,Inc. All Rights Reserved.

解説

P要素を利用する場合は段落の行揃えを、H1～H6要素を利用する場合は見出しの行揃えを、DIV要素を利用する場合は指定したブロックの範囲の行揃えをそれぞれ指定できます。

行揃え位置として指定する「left・right・center」は、それぞれ「左揃え・右揃え・中央揃え」を表しています。

ただし、align属性は将来的に廃止される予定となっていますので、行揃えの指定をする場合には、スタイルシートを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>行揃えサンプル</TITLE>
</HEAD>
<BODY>

<H1 align="center">夜明けの行揃え</H1>

<P>
実をいうと、行揃えには「left, right, center」のほかに
「justify」というものがある。いわゆる「両端揃え」のこと
である。しかし、色々試してはみたのだが、これが実際には・・・
</P>

<P align="right">【文:岡蔵　龍一】</P>

<DIV align="center">
<P>- ご感想をおよせください -</P>
<P>(C) Copyright 1999 OKARYU,Inc. All Rights Reserved.</P>
</DIV>

</BODY>
</HTML>
```

「テキスト」の「見出しを表す」:P.43参照
「テキスト」の「段落を表す」:P.45参照
「スタイルとレイアウト」の「センタリングする」:P.72参照
「スタイルシート」の「任意の範囲にスタイルを適用させるためのタグ」:P.194参照
「スタイルシート」の「行揃えを指定する」:P.207参照

HTML

センタリングする

<CENTER>～</CENTER>

Internet Explorer

センタリング

要素名	説明
CENTER	中央揃え

テキストはもちろん、画像もテーブルもOK!

Netscape Navigator

センタリング

要素名	説明
CENTER	中央揃え

テキストはもちろん、画像もテーブルもOK!

解説

指定したブロックの範囲の行揃えを「中央揃え」に設定します。
DIV要素のalign属性に「center」を指定した場合とまったく同じ働きをします。
ただし、この要素は将来的に廃止される予定となっていますので、スタイルシートを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>センタリングサンプル</TITLE>
</HEAD>
<BODY>
```

```
<CENTER>
<H1>センタリング</H1>

<IMG src="sample.gif" width="144" height="80" alt="">
<TABLE border="1">
<TR>
<TH>要素名</TH>
<TH>説明</TH>
</TR>
<TR>
<TD>CENTER</TD>
<TD>中央揃え</TD>
</TR>
</TABLE>
<P>テキストはもちろん、画像もテーブルもOK!</P>
</CENTER>
</BODY>
</HTML>
```

「スタイルとレイアウト」の「行揃えを指定する」:P.70参照
「スタイルシート」の「行揃えを指定する」:P.207参照

HTML

横罫線を入れる

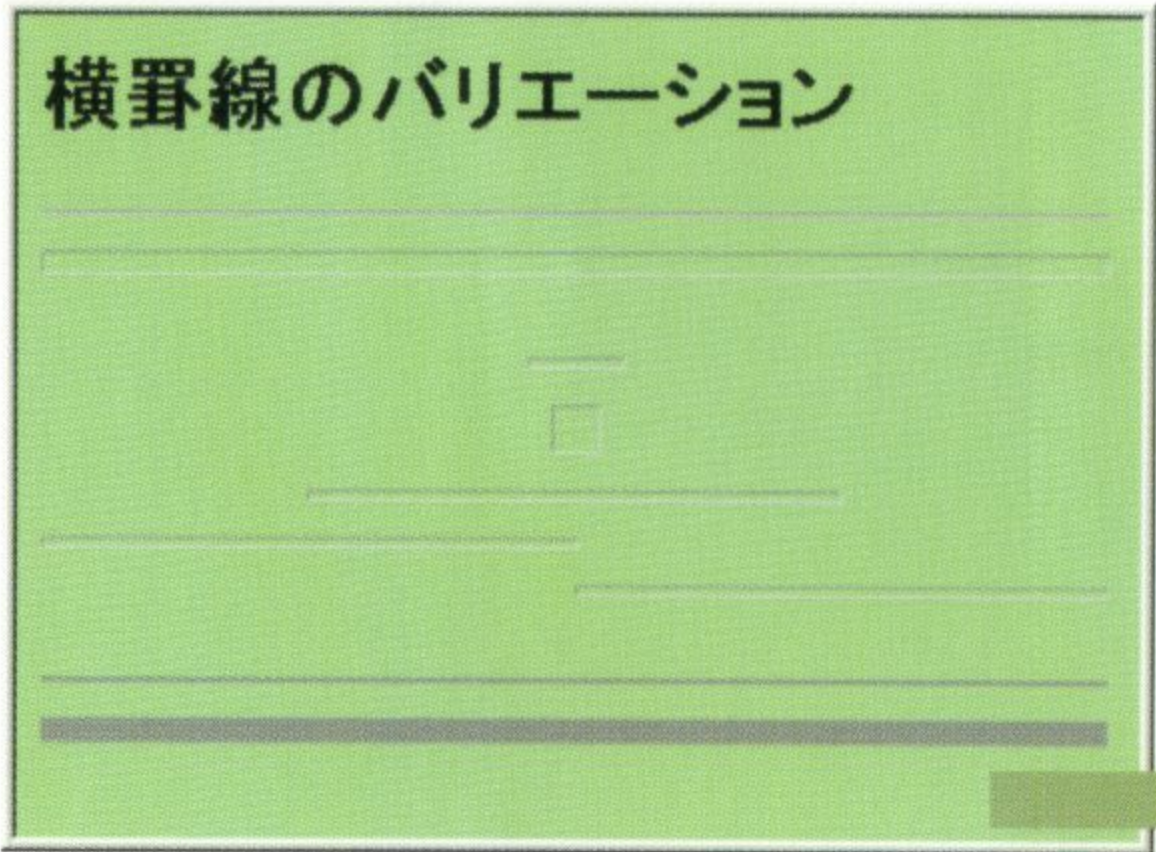
```
<HR>
<HR size="太さ" width="長さ" align="行揃え位置" noshade>
```

size	罫線の太さ(ピクセル)
width	罫線の長さ(ピクセルまたは%)
align	行揃え位置(left, right, center)
noshade	罫線を平面的に表示

Internet Explorer

横罫線のバリエーション

Netscape Navigator

横罫線のバリエーション

解説

この要素を入れると、そこに横罫線が引かれます。
一般的なブラウザでは、引っ込んだ感じで立体的に表示されますが、noshade属性を指定すると単純な塗りつぶしの線にすることができます。size属性は線の太さを、width属性は線の長さをピクセル単位で設定します。width属性は、値の単位として「%」をつけることで、ウインドウの幅に対する割合で長さを設定することもできます。align属性は、罫線の行揃えを設定しますが、何も指定しない場合は「center」になります。これらの属性は、必ずしもすべてを指定する必要はありませんので、必要なものだけを記述するようにしてください。
また、これらの属性はすべて、将来的に廃止されることになっています。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>横罫線サンプル</TITLE>
</HEAD>
<BODY bgcolor="#98FB98">

<H1>横罫線のバリエーション</H1>

<HR>
<HR size="10">
<BR>
<HR size="5"  width="40">
<HR size="20"  width="20">
<HR size="5"  width="50%">
<HR size="5"  width="50%" align="left">
<HR size="5"  width="50%" align="right">
<BR>
<HR noshade>
<HR noshade size="10">

</BODY>
</HTML>
```

HTML

他のページにリンクする

<A href="リンク先URL">〜</A>

Internet Explorer

詳解 HTML4.0 & Cascading Style Sheet 辞典
詳解 JavaScript & DynamicHTML 辞典

| トップ | 目次 | 前ページ | 次ページ |

Netscape Navigator

詳解 HTML4.0 & Cascading Style Sheet 辞典
詳解 JavaScript & DynamicHTML 辞典

| トップ | 目次 | 前ページ | 次ページ |

解説

指定した範囲を他のページにリンクするようにします。
リンクさせる部分の言葉には、「ここをクリック」などという言葉を使用せずに、リンク先を連想させるような意味のある具体的な言葉を使用した方がよいでしょう。
また、リンクする言葉が連続している場合には、その間にリンクしていない文字を入れるなどして明確に区切るようにしてください。
この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>リンクサンプル</TITLE>
</HEAD>
<BODY>
```

```
<P align="center">
<A href="books.html">
<IMG src="banner.gif" width="468" height="60"
 alt="関連書籍紹介" border="0"></A>
<BR>
  | <A href="index.html">トップ</A>
  | <A href="contents.html">目次</A>
  | <A href="section1">前ページ</A>
  | <A href="section3">次ページ</A>
  |
</P>
</BODY>
</HTML>
```


「ページの基礎となる内容」の「基本となる文字色を設定する」:P.32参照

「画像とマルチメディア」の「画像の枠線を設定する」:P.124参照

「スタイルシート」の「リンク部分のスタイルを指定する」:P.228参照

「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照

「アクセシビリティ」の「ショートカットキーを割り当てる」:P.241参照

コラム

新しいウインドウを開くことは便利なこと？

普通にリンクされているのであれば、リンク先を同じウインドウに開くことも、新しいウインドウに開くこともできます(Windowsの場合は右クリック、Macintoshの場合はプレスして表示されるメニューで選択できる)。つまり、ユーザーが、その時の状況に応じて選択できるのです。

しかし、target属性で新しいウインドウに表示するように設定されたリンクは、現在のところ、同じウインドウに表示させることはできません。ユーザーの意思とは無関係に、常に新しいウインドウに表示されてしまいます。

ユーザーの中には、メモリが少なくて多くのウインドウを開けない状況の人や、新しいウインドウが開かれたことを目で確認できないユーザーもいます。そのような意味で、アクセシビリティのガイドラインでは、「あらかじめユーザーに知らせることなしに、新しいウインドウを開かないようにする」ことが推奨されています。

HTML

同じページ内の指定した位置へリンクする

<A name="位置名">～</A>　←リンク先の指定
<A href="#位置名">～</A>　←リンク元の指定

Internet Explorer

アクセシブルなウェブデザイン

目次

- 1. はじめに
- 2. 同等の意味を持つ代わりのテキストを提供する
- 3. 色に依存しない
- 4. 正しくタグ付けし、適切にスタイルシートを使う
- 5. 点滅や移動をするテキストは止められるようにする
- 6. 装置に依存しないように設計する
- 7. ブラウザが対応していない場合は、暫定的な解決策をとる
- 8. 前後関係や位置を表す情報を提供する
- 9. はっきりとわかるナビゲーションのための仕組みを提供する

---

1. はじめに

【目次 | 次の項目】

Webページデザインに関連するアクセシビリティについてよく知らない方は、多くのユーザーがあなたとは非常に異なった状況の下で操作している可能性があるということを考えてみてください。

- あるユーザーは、「見ることができない」「聞くことができない」「動くことができない」または「ある種類の情報を簡単に、あるいはまったく処理できない」かもしれません。
- あるユーザーは、「キーボードやマウスがない」または「キーボードやマウスを使うことができない」かもしれません。
- あるユーザーは、「テキストしか表示できない環境」「小さな画面を使用」「インターネットに低速でしか接続できない環境」で操作しているかもしれません。
- あるユーザーは、「見たり聞いたりできない状況」または「手が使えない状況」にあるかもしれません(車を運転している場合や、騒がしい環境などの場合)。
- あるユーザーは、「古いバージョンのブラウザ」「まったく異なる種類のブラウザ」「音声出力のブラウザ」「異なるOS」などを使用しているかもしれません。

2. 同等の意味を持つ代わりのテキストを提供する

【前の項目 | 目次 | 次の項目】

Netscape Navigator

アクセシブルなウェブデザイン

目次

- 1. はじめに
- 2. 同等の意味を持つ代わりのテキストを提供する
- 3. 色に依存しない
- 4. 正しくタグ付けし、適切にスタイルシートを使う
- 5. 点滅や移動をするテキストは止められるようにする
- 6. 装置に依存しないように設計する
- 7. ブラウザが対応していない場合は、暫定的な解決策をとる
- 8. 前後関係や位置を表す情報を提供する
- 9. はっきりとわかるナビゲーションのための仕組みを提供する

1. はじめに

【目次 | 次の項目】

Webページデザインに関連するアクセシビリティについてよく知らない方は、多くのユーザーがあなたとは非常に異なった状況の下で操作している可能性があるということを考えてみてください。

- あるユーザーは、「見ることができない」「聞くことができない」「動くことができない」または「ある種類の情報を簡単に、あるいはまったく処理できない」かもしれません。
- あるユーザーは、「キーボードやマウスがない」または「キーボードやマウスを使うことができない」かもしれません。
- あるユーザーは、「テキストしか表示できない環境」「小さな画面を使用」「インターネットに低速でしか接続できない環境」で操作しているかもしれません。
- あるユーザーは、「見たり聞いたりできない状況」または「手が使えない状況」にあるかもしれません(車を運転している場合や、騒がしい環境などの場合)。
- あるユーザーは、「古いバージョンのブラウザ」「まったく異なる種類のブラウザ」「音声出力のブラウザ」「異なるOS」などを使用しているかもしれません。

2. 同等の意味を持つ代わりのテキストを提供する

【前の項目 | 目次 | 次の項目】

目次の「同等の意味を持つかわりのテキストを提供する」をクリックすると、<A name="epuiv">と指定されている部分にジャンプします。

2. 同等の意味を持つ代わりのテキストを提供する

【前の項目 | 目次 | 次の項目】

画像や映像、音、アプレットなどに関しては直接利用することができない人もいますが、それらに対して同等の意味を持つ情報も含まれていれば、そのページを利用することが可能になります。この場合、同等の意味を持つ情報は、見るための内容や聞くための内容と同じ効果を持っていなければなりません。したがって、例えば、目次にリンクしている「上向きの矢印の画像」に対する同等の意味を持つテキストの場合は、「目次へ」のようになります。場合によっては、同等の意味を持つ代わりのものは、見るための内容そのままの姿や聞くための内容の音そのものを表現すべき時もあります。(例: 複雑な図、広告、図形や教育のために利用される音声のサンプルなど)

2. 同等の意味を持つ代わりのテキストを提供する

【前の項目 | 目次 | 次の項目】

画像や映像、音、アプレットなどに関しては直接利用することができない人もいますが、それらに対して同等の意味を持つ情報も含まれていれば、そのページを利用することが可能になります。この場合、同等の意味を持つ情報は、見るための内容や聞くための内容と同じ効果を持っていなければなりません。したがって、例えば、目次にリンクしている「上向きの矢印の画像」に対する同等の意味を持つテキストの場合は、「目次へ」のようになります。場合によっては、同等の意味を持つ代わりのものは、見るための内容そのままの姿や聞くための内容の音そのものを表現すべき時もあります。(例: 複雑な図、広告、図形や教育のために利用される音声のサンプルなど)

3. 色に依存しない

解説

1つのページがとても長い場合などに、同じページ内の特定の位置に名前を付けておいて、そこにリンク(ジャンプ)することができます。
リンクの対象となる位置に名前を付けるには、name属性を使用します。そして、そこへリンクするためには、href属性でリンク先の名前の前に「#」記号を付けて指定します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>アクセシブルなウェブデザイン</TITLE>
</HEAD>
<BODY bgcolor="white">

<H1>アクセシブルなウェブデザイン</H1>

<H2><A name="contents">目次</A></H2>
<UL>
<LI><A href="#intro">1. はじめに</A>
<LI><A href="#equiv">2. 同等の意味を持つ代わりのテキストを提供する</A>
<LI><A href="#color">3. 色に依存しない</A>
<LI><A href="#strct">4. 正しくタグ付けし、適切にスタイルシートを使う</A>
<LI><A href="#mvmnt">5. 点滅や移動をするテキストは止められるようにする
</A>
<LI><A href="#indep">6. 装置に依存しないように設計する</A>
<LI><A href="#intrm">7. ブラウザが対応していない場合は、暫定的な解決策
をとる</A>
<LI><A href="#cmplx">8. 前後関係や位置を表す情報を提供する </A>
<LI><A href="#navig">9. はっきりとわかるナビゲーションのための仕組みを
提供する</A>
</UL>

<HR>

<H2><A name="intro">1. はじめに</A></H2>
<P>
【<A href="#contents">目次</A>
 | <A href="#equiv">次の項目</A>】
</P>
<P>
Web ページデザインに関連するアクセシビリティ（中略）考えてみてください。
</P>

<UL>
<LI> あるユーザーは、（中略）かもしれません。
<LI> あるユーザーは、（中略）かもしれません。
<LI> あるユーザーは（中略）かもしれません。
```

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML

```
<LI> あるユーザーは（中略）(車を運転している場合や、騒がしい環境などの場合)。
<LI> あるユーザーは、（中略）使用しているかもしれません。
</UL>

<H2><A name="equiv">2. 同等の意味を持つ代わりのテキストを提供する</A>
</H2>
<P>
【<A href="#intro">前の項目</A>
 | <A href="#contents">目次</A>
 | <A href="#color">次の項目</A>】
</P>
<P>
画像や映像、音、アプレットなどに（中略）音声のサンプルなど)
</P>

<H2><A name="color">3. 色に依存しない</A></H2>
<P>
【<A href="#equiv">前の項目</A>
 | <A href="#contents">目次</A>
 |<A href="#strct">次の項目</A>】
</P>
<P>
ある情報を伝えるために（中略）十分なコントラストをあたえることができません。
</P>

<H2><A name="strct">4. 正しくタグ付けし、適切にスタイルシートを使う</A>
</H2>
<P>
【<A href="#color">前の項目</A>
 | <A href="#contents">目次</A>
 | <A href="#mvmnt">次の項目</A>】
</P>
<P>
タグ付けを正しく行わない(仕様に従わない)ことは、（中略）表現することを難しくします。
</P>

</BODY>
</HTML>
```

HTML

他のページ内の指定した位置へリンクする

```
<A name="位置名">～</A>            ←リンク先の指定
<A href="URL#位置名">～</A>        ←リンク元の指定
```

Internet Explorer

Web Page Design Univ.

第3回 アクセシビリティの重要性

さて、今回はアクセシビリティについて考えてみましょう。アクセシビリティというと、「あ～、画像にALTを付けることでしょう」と答える方も多いようです。アクセシビリティとは決してそれだけのものではないのですが、それは「同等の意味を持つ代わりのテキストを提供する」という点で重要なことでもあります。

2. 同等の意味を持つ代わりのテキストを提供する

【前の項目 | 目次 | 次の項目】

画像や映像、音、アプレットなどに関しては直接利用することができない人もいますが、それらに対して同等の意味を持つ情報も含まれ

Netscape Navigator

Web Page Design Univ.

第3回 アクセシビリティの重要性

さて、今回はアクセシビリティについて考えてみましょう。アクセシビリティというと、「あ～、画像にALTを付けることでしょう」と答える方も多いようです。アクセシビリティとは決してそれだけのものではないのですが、それは「同等の意味を持つ代わりのテキストを提供する」という点で重要なことでもあります。

2. 同等の意味を持つ代わりのテキストを提供する

【前の項目 | 目次 | 次の項目】

画像や映像、音、アプレットなどに関しては直接利用することができない人もいますが、それらに対して同等の意味を持つ情報も含まれていれば、そのページを利用することが可能になります。この場合、

解説

他のページにリンクする場合に、そのページ内の特定の位置に名前を付けておいて、その位置にリンク(ジャンプ)させることができます。
リンクの対象となる位置に名前を付けるには、name属性を使用します。そして、その位置へリンクするためには、href属性にURL+「#」記号+リンクする位置の名前を指定します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>Web Page Design Univ.</TITLE>
</HEAD>
<BODY>
<H1>Web Page Design Univ.</H1>
<H2>第3回　アクセシビリティの重要性</H2>
<P>
さて、今回はアクセシビリティについて（中略）ないのですが、それは「<A href="
../accessibility/index.html#equiv">同等の意味を持つ代わりのテキス
トを提供する</A>」という点で重要なことでもあります。
</P>
</BODY>
</HTML>
```

リンク先のソースについては、「同じページ内の指定した位置へリンクする」(P.78)を参照してください。

HTML リンク先を別のウインドウに表示する

```
<A href="URL" target="ウインドウ名">~</A>
```

Internet Explorer

オンライン絵本・リンク集

● べこいぬ

かわいい小犬の「べこいぬ」が、知らないうちに生態系(食物連鎖)を壊してしまうという、ちょっと変わったお話です。意外な展開をお楽しみください。

べこいぬ

Netscape Navigator

オンライン絵本・リンク集

● べこいぬ

かわいい小犬の「べこいぬ」が、知らないうちに生態系(食物連鎖)を壊してしまうという、ちょっと変わったお話です。意外な展開をお楽しみください。

べこいぬ

解説

target属性を利用すると、リンク先のページを表示させるウインドウを指定することができます。
すでに指定した名前のウインドウがある場合はそのウインドウへ、ない場合は新しいウインドウに表示します。
この時、新しく開いたウインドウの名前は、target属性で指定したものになります。
また、「_blank」のように「_」で始まる特別な名前がいくつかあります。「_blank」は新しいウインドウを名前のない状態で開き、「_self」はリンク元と同じウインドウに表示させます。

Sample

```
<H1>オンライン絵本・リンク集</H1>
<H2>●<A href="http://www.phoenix-c.or.jp/~zspc/bekoinu
/" target="ehon">
べこいぬ</A>
</H2>
<P>
かわいい小犬の「べこいぬ」が、知らないうちに生態系(食物連鎖)を壊してしまうという、
ちょっと変わったお話です。意外な展開をお楽しみください。
</P>
```

Tip「ターゲット名について」:P.179参照

HTML

リンクでメールを送れるようにする

<A href="mailto:*メールアドレス*">～</A>

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

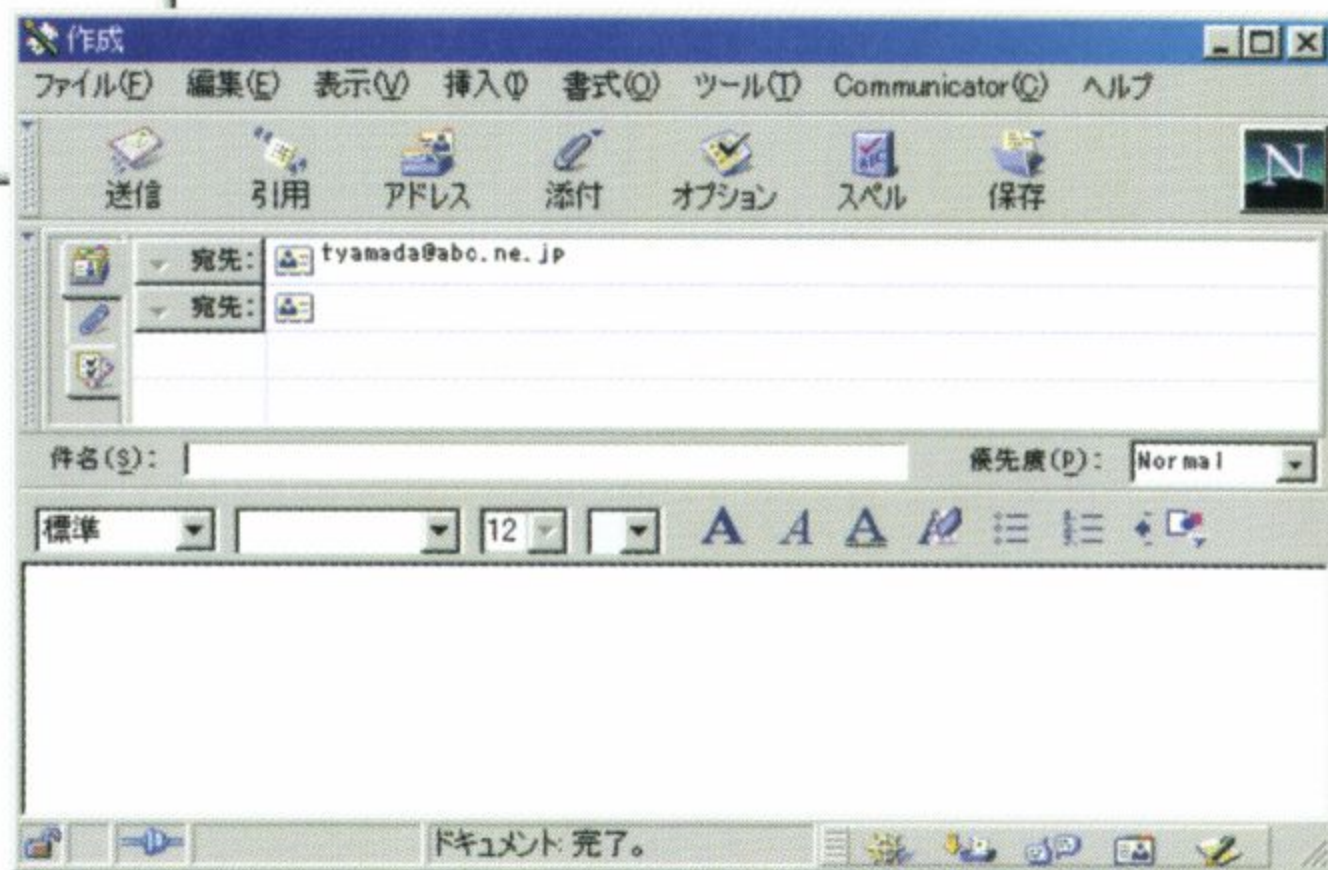
Internet Explorer

ご意見・ご感想をお待ちしています。

山田太郎 tyamada@abc.ne.jp

Netscape Navigator

ご意見・ご感想をお待ちしています。

山田太郎 tyamada@abc.ne.jp

解説

リンクの通常URLを記入する部分に、「mailto:」に続けてメールアドレスを記入しておくと、ブラウザのメールを送信するウインドウを開いたり、設定されているメールソフトを起動することができます。

この場合、指定したメールアドレスが自動的に入力された状態になります。

この機能は、必ずしもすべてのブラウザで利用できるわけではありませんので、メールアドレスが画面に表示されるようにしておく(つまり、メールアドレスをリンクする文字に含める)とよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>リンクでメールを送れるようにする</TITLE>
</HEAD>
<BODY>
<P>
ご意見・ご感想をお待ちしています。
</P>
<ADDRESS>
山田太郎 <A href="mailto:tyamada@abc.ne.jp">
tyamada@abc.ne.jp</A>
</ADDRESS>
</BODY>
</HTML>
```

「ページの基礎となる内容」の「内容に関する問い合わせ先を示す」:P.41参照

HTML

マーク付きのリストを作る

<UL><LI> *リスト項目1*</LI><LI> *リスト項目2*</LI>・・・</UL>

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

Internet Explorer

関連書籍

- 詳解HTML&JavaScript辞典
- 詳解HTML4.0&CascadingStyleSheet辞典
- 詳解JavaScript&DynamicHTML辞典

Netscape Navigator

関連書籍

- 詳解HTML&JavaScript辞典
- 詳解HTML4.0&CascadingStyleSheet辞典
- 詳解JavaScript&DynamicHTML辞典

Lynx

AUTO マークつきリスト・サンプル

関連書籍
* 詳解HTML&JavaScript辞典
* 詳解HTML4.0&CascadingStyleSheet辞典
* 詳解JavaScript&DynamicHTML辞典

解説

連番付きではなく、丸や四角などのマーク付きのリストを作ります。
<UL>(Unordered List)はその範囲全体が順不同リストであることを示し、<LI>(List Item)はリスト内の各項目を表します。したがって、書き方としては、リスト全体を<UL>～</UL>で囲み、リスト内の各項目は<LI>タグで囲って示します。この時、<LI>の終了タグである</LI>は省略することができます。
一般的なブラウザでは、全体にインデントされた状態で、各項目は自動的に改行されて表示されます。また、マークとしては黒丸が一般的ですが、すべてのブラウザでそのように表示されるとは限りません。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>マーク付きリスト・サンプル</TITLE>
</HEAD>
<BODY>
<P>関連書籍</P>
<UL>
<LI>詳解HTML&JavaScript辞典</LI>
<LI>詳解HTML4.0&CascadingStyleSheet辞典</LI>
<LI>詳解JavaScript&DynamicHTML辞典</LI>
</UL>
</BODY>
</HTML>
```

HTML リストのマークを変える

```
<UL type="マークの種類">〜</UL>
<LI type="マークの種類">〜</LI>
```

【マークの種類】

disc	黒まる
circle	白まる
square	四角

Internet Explorer

関連書籍

- ● 詳解HTML&JavaScript辞典
- ○ 詳解HTML4.0&CascadingStyleSheet辞典
- ■ 詳解JavaScript&DynamicHTML辞典

Netscape Navigator

関連書籍

- ・ 詳解HTML&JavaScript辞典
- ・ 詳解HTML4.0&CascadingStyleSheet辞典
- ■ 詳解JavaScript&DynamicHTML辞典

解説

マーク付きリストの各項目の前に表示されるマークの種類を設定します。<UL>に指定した場合はリスト全体がそのマークに変更され、<LI>に指定した場合はその項目のみが変更されます。ただし、これらのマークはブラウザの種類によって、その大きさや色(squareには白抜きのものと塗りつぶしのものがあります)などが異なります。また、このtype属性によってマークを変更する方法は将来的に廃止される予定となっていますので、代わりにスタイルシートを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>マークの種類・サンプル</TITLE>
</HEAD>
<BODY>
<P>関連書籍</P>
<UL>
<LI type="disc">詳解HTML&JavaScript辞典</LI>
<LI type="circle">詳解HTML4.0&CascadingStyleSheet辞典</LI>
<LI type="square">詳解JavaScript&DynamicHTML辞典</LI>
</UL>
</BODY>
</HTML>
```

「スタイルシート」の「リストのマークや番号の形式を変える」:P.233参照

番号付きのリストを作る

<OL><LI> *リスト項目1*</LI><LI> *リスト項目2*</LI>・・・</OL>

Internet Explorer

関連書籍

1. 詳解HTML&JavaScript辞典
2. 詳解HTML4.0&CascadingStyleSheet辞典
3. 詳解JavaScript&DynamicHTML辞典

Netscape Navigator

関連書籍

1. 詳解HTML&JavaScript辞典
2. 詳解HTML4.0&CascadingStyleSheet辞典
3. 詳解JavaScript&DynamicHTML辞典

解説

マーク付きではなく、数字の連番付きのリストを作ります。
<OL>(Ordered List)はその範囲全体が連番付きリストであることを示し、<LI>(List Item)はリスト内の各項目を表します。したがって、書き方としては、リスト全体を<OL>～</OL>で囲み、リスト内の各項目は<LI>タグで囲って示します。この時、<LI>の終了タグである</LI>は省略することができます。
一般的なブラウザでは、全体にインデントされた状態で、各項目は自動的に改行されて表示されます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>番号付きリスト・サンプル</TITLE>
</HEAD>
<BODY>
<P>関連書籍</P>
<OL>
<LI>詳解HTML&JavaScript辞典</LI>
<LI>詳解HTML4.0&CascadingStyleSheet辞典</LI>
<LI>詳解JavaScript&DynamicHTML辞典</LI>
</OL>
</BODY>
</HTML>
```

「スタイルシート」の「リストのマークや番号の形式を変える」：P.233参照

HTML

リストの番号の形式を変える

<OL type="番号の形式">～</OL>
<LI type="番号の形式">～</LI>

【番号の形式】

1	算用数字	1・2・3…
a	英小文字	a・b・c…
A	英大文字	A・B・C…
i	ローマ数字小文字	i・ii・iii…
I	ローマ数字大文字	I・II・III…

Internet Explorer

関連書籍

i. 詳解HTML&JavaScript辞典
ii. 詳解HTML4.0&CascadingStyleSheet辞典
iii. 詳解JavaScript&DynamicHTML辞典

Netscape Navigator

関連書籍

i. 詳解HTML&JavaScript辞典
ii. 詳解HTML4.0&CascadingStyleSheet辞典
iii. 詳解JavaScript&DynamicHTML辞典

解説

番号付きリストの各項目の前に表示される番号の形式を設定します。
<OL>に指定した場合はリスト全体の番号がその形式に変更され、<LI>に指定した場合はその項目のみが変更されます。
ただし、このtype属性によって番号の形式を変更する方法は、将来的に廃止される予定となっていますので、代わりにスタイルシートを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>番号の形式・サンプル</TITLE>
</HEAD>
<BODY>
<P>関連書籍</P>
<OL type="i">
<LI>詳解HTML&JavaScript辞典</LI>
<LI>詳解HTML4.0&CascadingStyleSheet辞典</LI>
<LI>詳解JavaScript&DynamicHTML辞典</LI>
</OL>
</BODY>
</HTML>
```

「スタイルシート」の「リストのマークや番号の形式を変える」:P.233参照

HTML

リストの連番を変更する

```
<OL start="開始番号">~</OL>
<LI value="開始番号">~</LI>
```

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

Internet Explorer

4. リスト項目A
5. リスト項目B
6. リスト項目C
1. リスト項目D
2. リスト項目E
3. リスト項目F

Netscape Navigator

4. リスト項目A
5. リスト項目B
6. リスト項目C
1. リスト項目D
2. リスト項目E
3. リスト項目F

解説

番号付きリストの番号は1から開始されますが、start属性を指定すると任意の番号から開始させることができます。
value属性も同様に開始番号を変更しますが、こちらはリスト項目<LI>に対して設定しますので、連番を途中から変更することができます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>連番の変更・サンプル</TITLE>
</HEAD>
<BODY>
<OL start="4">
<LI>リスト項目A</LI>
<LI>リスト項目B</LI>
<LI>リスト項目C</LI>
<LI value="1">リスト項目D</LI>
<LI>リスト項目E</LI>
<LI>リスト項目F</LI>
</OL>
</BODY>
</HTML>
```

HTML

定義形式リスト(用語とその説明)を作る

<DL><DT>定義する言葉</DT><DD>その説明</DD>・・・</DL>
<DL><DT>定義する言葉<DD>その説明・・・</DL>

Internet Explorer

ユーザーエージェント
ウェブ・コンテンツにアクセスするためのソフトウェア。一般的なデスクトップのグラフィカルなブラウザの他、音声ブラウザ、携帯電話、マルチメディアプレイヤー、プラグインなども含まれる。

Netscape Navigator

ユーザーエージェント
ウェブ・コンテンツにアクセスするためのソフトウェア。一般的なデスクトップのグラフィカルなブラウザの他、音声ブラウザ、携帯電話、マルチメディアプレイヤー、プラグインなども含まれる。

解説

定義形式リストとは、定義する言葉とそれに対する説明文を対にした形式のリストです。
<DL>～</DL>の範囲には、DT要素とDD要素を1組だけでなく、それぞれ必要な数だけ任意の順序で入れることができます。また、<DT>と<DD>の終了タグである</DT>と</DD>は省略することが可能です。
一般的なブラウザでは、<DD>～</DD>の範囲がインデントされて表示されます。そのため、かつてはマージンをとる目的で利用されることも多かったのですが、現在ではスタイルシートで自由にマージンを設定することができますので、そちらを利用するようにしてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>定義形式リスト・サンプル</TITLE>
</HEAD>
<BODY>
<DL>
<DT>ユーザーエージェント</DT>
<DD>
ウェブ・コンテンツにアクセスするためのソフトウェア。(中略) プラグインなども含まれる。
</DD>
</DL>
</BODY>
</HTML>
```

「スタイルシート」の「マージンを設定する」:P.205参照

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML

表の基本形

<TABLE border="外枠の太さ">～</TABLE> ←表全体
<TR>～</TR> ←横1列
<TH>～</TH> ←見出し用セル
<TD>～</TD> ←データ用セル

外枠の太さ　ピクセル

Internet Explorer

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

Netscape Navigator

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

解説

表は、その全体を<TABLE>～</TABLE>で囲って示します。
各セルは、それが列や行に対する見出しの役割をするのであれば<TH>～</TH>で、データであれば<TD>～</TD>で囲って示します。
一般的なブラウザでは、border属性を指定しなければ表の枠は表示されません。
また、見出し用のセル内の文字は太字でセンタリングされて表示されます。横1列分のセルは、<TR>～</TR>で囲います。

Sample

```
<BODY>
<TABLE border="2">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>
</BODY>
```

表を作る

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

HTML

表にタイトルを付ける

```
<CAPTION>～</CAPTION>
<CAPTION align="表示位置">～</CAPTION>
```

表示位置　　top・bottom・(left・right)

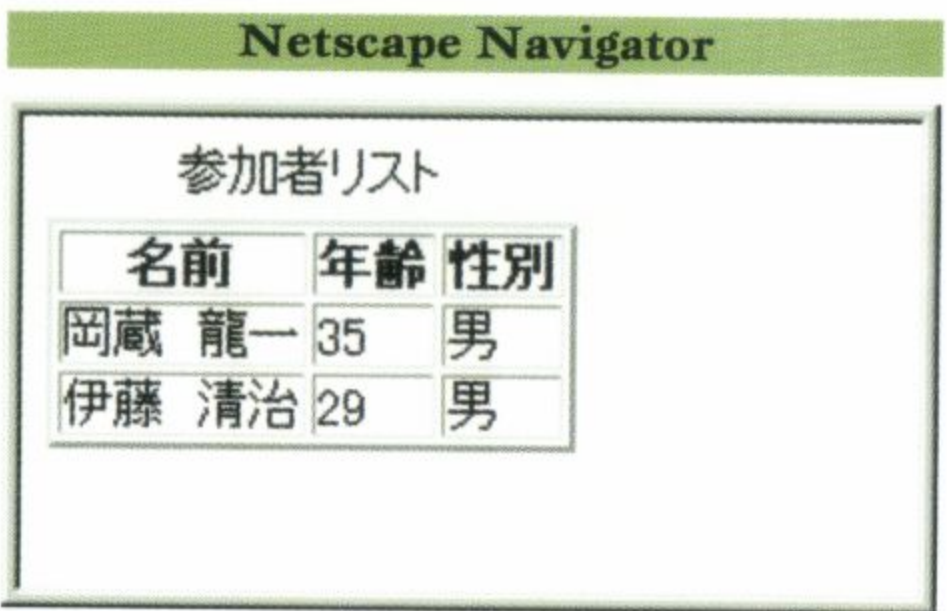
Internet Explorer

参加者リスト

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

Netscape Navigator

参加者リスト

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

解説

表にタイトル(キャプション)を付けます。
align属性で表示位置を指定しない場合には、テーブルの上に表示されます。
一般的なブラウザでは、表に対してセンタリングされて表示されます。
表示位置のleftとrightに関しては、正しく対応していないブラウザが多いので、使用しない方がよいでしょう(align属性自体も廃止される予定となっています)。この要素を使用する場合は、必ず<TABLE>タグの直後に配置してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>表タイトル・サンプル</TITLE>
</HEAD>
<BODY>
<TABLE border="2">
<CAPTION>参加者リスト</CAPTION>
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

表の大きさを指定する

<TABLE width="*幅*" height="*高さ*">～</TABLE>

幅・高さ　　ピクセル数またはウインドウに対する%

Internet Explorer

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

Netscape Navigator

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

解説

表全体の幅と高さを指定します。
一般的なブラウザの多くはheight属性をサポートしていますが、HTML4.0ではTABLE要素に対してこの属性は定義されていません。表のセルを表すTH要素とTD要素ではheight属性が利用できますので、高さを指定したい場合はそちらを利用するようにしてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>表の大きさの指定サンプル</TITLE>
</HEAD>
<BODY>
<TABLE border="2" width="100%">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

「テーブル」の「セルの大きさを指定する」:P.113参照

HTML

表の枠やマージンなどを指定する

<TABLE border="*外枠の太さ*">〜</TABLE>
<TABLE cellspacing="*セルの間隔*">〜</TABLE>
<TABLE cellpadding="*セル内のマージン*">〜</TABLE>

外枠の太さ	ピクセル
セルの間隔	ピクセル
セル内のマージン	ピクセル

Internet Explorer

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

Netscape Navigator

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

Internet Explorer

■ border属性

border="0"　border="5"　border="10"

名前	年齢
岡蔵 龍一	35

■ cellspacing属性

cellspacing="0"　cellspacing="5"　cellspacing="10"

名前	年齢
岡蔵 龍一	35

■ cellpadding属性

cellpadding="0"　cellpadding="5"　cellpadding="10"

名前	年齢
岡蔵 龍一	35

Netscape Navigator

■ border属性

border="0"　border="5"　border="10"

名前	年齢
岡蔵 龍一	35

■ cellspacing属性

cellspacing="0"　cellspacing="5"　cellspacing="10"

名前	年齢
岡蔵 龍一	35

■ cellpadding属性

cellpadding="0"　cellpadding="5"　cellpadding="10"

名前	年齢
岡蔵 龍一	35

解説

border属性は、表全体の上下左右の外枠の太さを設定します。
値として0を指定すると、枠が表示されなくなります。
cellspacing属性は、各セルとセル及びセルと外枠の間隔を設定します。cellpadding属性は、セル内のマージン(セルの内容と枠との間隔)を設定します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>枠やマージン・サンプル</TITLE>
</HEAD>
<BODY>
<TABLE border="8" cellspacing="4" cellpadding="10">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

表の外枠の色を指定する

HTML

```
<TABLE bordercolor="色">～</TABLE>
<TABLE bordercolorlight="色" bordercolordark="色">～</TABLE>
```

Internet Explorer

bordercolor

名前	年齢	性別
岡蔵　龍一	35	男

bordercolorlight, bordercolordark

名前	年齢	性別
岡蔵　龍一	35	男

Netscape Navigator

bordercolor

名前	年齢	性別
岡蔵　龍一	35	男

bordercolorlight, bordercolordark

名前	年齢	性別
岡蔵　龍一	35	男

解説

bordercolor属性は、表の外枠全体の色を設定します。
Internet Explorerでは、ここで指定した色で外枠が平面的に塗りつぶされ、Netscape Navigatorでは、この色調で立体的に表示されます。
bordercolorlight属性とbordercolordark属性はInternet Explorerでのみ利用可能で、それぞれ立体的な外枠の明るい部分と暗い部分の色を設定します。これらの属性は、各社ブラウザの独自拡張でHTML4.0では定義されていないものです。表の枠の色を指定したい場合には、スタイルシートを利用するのが正しいやり方です(ただし、多くのブラウザは、まだサポートしていません)。

Sample

```
<HTML lang="ja">
<HEAD>
<TITLE>表の外枠の色サンプル</TITLE>
</HEAD>
<BODY>

<P>bordercolor</P>
<TABLE border="8" bordercolor="#0000FF">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
```

```
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TABLE>

<P>bordercolorlight, bordercolordark</P>
<TABLE border="8" bordercolorlight="#FF9900" bordercolor
dark="#CC0000">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TABLE>

</BODY>
</HTML>
```

「HTMLについて」の「色の指定方法」:P.25参照
「スタイルシート」の「枠の色を指定する」:P.222参照
巻末付録「カラーチャート1～3」:巻末参照

IE3.0
IE4.0
IE5.0

表の外枠の表示形式を指定する

HTML

<TABLE frame="外枠の表示形式">～</TABLE>

【外枠の表示形式】

void	なし(デフォルト)
above	上のみ
below	下のみ
lhs	左のみ
rhs	右のみ
hsides	上下のみ
vsides	左右のみ
box	上下左右
border	上下左右

Internet Explorer

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

Netscape Navigator

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

Internet Explorer

■ frame属性

frame="void"

名前	年齢
岡蔵 龍一	35

frame="above"

名前	年齢
岡蔵 龍一	35

frame="below"

名前	年齢
岡蔵 龍一	35

frame="lhs"

名前	年齢
岡蔵 龍一	35

frame="rhs"

名前	年齢
岡蔵 龍一	35

frame="hsides"

名前	年齢
岡蔵 龍一	35

frame="vsides"

名前	年齢
岡蔵 龍一	35

frame="box"

名前	年齢
岡蔵 龍一	35

frame="border"

名前	年齢
岡蔵 龍一	35

Netscape Navigator

■ frame属性

frame="void"

名前	年齢
岡蔵 龍一	35

frame="above"

名前	年齢
岡蔵 龍一	35

frame="below"

名前	年齢
岡蔵 龍一	35

frame="lhs"

名前	年齢
岡蔵 龍一	35

frame="rhs"

名前	年齢
岡蔵 龍一	35

frame="hsides"

名前	年齢
岡蔵 龍一	35

frame="vsides"

名前	年齢
岡蔵 龍一	35

frame="box"

名前	年齢
岡蔵 龍一	35

frame="border"

名前	年齢
岡蔵 龍一	35

解説 frame属性は、表全体を囲う外枠のうち、上下左右のどの枠を表示させるかを設定します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>表の外枠の表示形式サンプル</TITLE>
</HEAD>
<BODY>
<TABLE border="6" frame="hsides">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

IE3.0
IE4.0
IE5.0

HTML

表内のセルを区切る線の表示形式を指定する

<TABLE rules="*セルを区切る線の表示形式*">～</TABLE>

【セルを区切る線の表示形式】

none	なし(デフォルト)
rows	横列の境界のみ
cols	縦列の境界のみ
groups	THEAD・TBODY・TFOOT・COLGROUP・COLの境界のみ
all	すべての境界

Internet Explorer

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

■ rules属性

rules="none" rules="rows" rules="cols" rules="groups" rules="all"

Netscape Navigator

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

■ rules属性

rules="none" rules="rows" rules="cols" rules="groups" rules="all"

表を作る

解説

rules属性は、セルを区切る線の内、どの線を表示させるかを設定します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>セルを区切る線の表示形式サンプル</TITLE>
</HEAD>
<BODY>
<TABLE border="6" rules="cols">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

「テーブル」の「横列をグループ化する」:P.118参照
「テーブル」の「縦列をグループ化する」:P.120参照
「テーブル」の「縦列に幅や行揃えをまとめて指定する」:P.122参照

NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML

表の背景色を指定する

```
<TABLE bgcolor="色">～</TABLE>     ←表全体の背景色
<TR bgcolor="色">～</TR>           ←横1列の背景色
<TH bgcolor="色">～</TH>           ←見出し用セルの背景色
<TD bgcolor="色">～</TD>           ←データ用セルの背景色
```

Internet Explorer

名前	年齢	性別
岡蔵　龍一	35	男
山田　花子	22	女
伊藤　清治	29	男

Netscape Navigator

名前	年齢	性別
岡蔵　龍一	35	男
山田　花子	22	女
伊藤　清治	29	男

解説

bgcolor属性を使用すると、背景色を設定することができます。
<TABLE>に対して指定すると表全体の背景に、<TR>に対して指定するとその横1列の背景に、<TH>と<TD>ではそのセルの背景に対して色が付きます。
ただし、この属性は廃止予定となっていますので、スタイルシートを使った方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>表の背景色サンプル</TITLE>
</HEAD>
<BODY bgcolor="#CCCCCC">
<TABLE border="2" cellpadding="8">
<TR>
<TH>名前</TH>
<TH bgcolor="#999999">年齢</TH>
<TH>性別</TH>
</TR>
```

表を作る

```
<TR>
<TD bgcolor="#0099CC">岡蔵　龍一</TD>
<TD>35</TD>
<TD bgcolor="#0099CC">男</TD>
</TR>
<TR bgcolor="#FFCCCC">
<TD>山田　花子</TD>
<TD>22</TD>
<TD>女</TD>
</TR>
<TR>
<TD bgcolor="#0099CC">伊藤　清治</TD>
<TD>29</TD>
<TD bgcolor="#0099CC">男</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

「HTMLについて」の「色の指定方法」：P.25参照
「スタイルシート」の「背景色を指定する」：P.196参照
巻末付録「カラーチャート1～3」：巻末参照

NN4.0 IE3.0 IE4.0 IE5.0

HTML

表の背景画像を指定する

```
<TABLE background="画像のURL">～</TABLE>   ←表全体の背景画像
<TR background="画像のURL">～</TR>         ←横1列の背景画像
<TH background="画像のURL">～</TH>         ←見出し用セルの背景画像
<TD background="画像のURL">～</TD>         ←データ用セルの背景画像
```

Internet Explorer

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

Netscape Navigator

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

解説

background属性を使用すると、背景画像を設定することができます。<TABLE>に対して指定すると表全体の背景に、<TR>に対して指定するとその横1列の背景に、<TH>と<TD>ではそのセルの背景に対して背景画像が表示されます。画像ファイルとしては、一般にGIF形式の他、JPEG形式やPNG形式も使用できます。ただし、この属性は一部ブラウザの独自拡張で、HTML4.0では定義されていないものです。表に背景画像を指定したい場合には、スタイルシートを利用するのが正しいやり方です。

Sample

```
<HTML lang="ja">
<HEAD>
<TITLE>表の背景画像サンプル</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<TABLE border="2" cellpadding="15">
<TR background="green.gif">
<TH>名前</TH>
<TH>年齢</TH>
<TH>性別</TH>
</TR>
<TR background="paper.gif">
```

表を作る

```
<TD>岡蔵　龍一</TD>
<TD>35</TD>
<TD>男</TD>
</TR>
<TR background="paper.gif">
<TD>伊藤　清治</TD>
<TD>29</TD>
<TD>男</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

「スタイルシート」の「背景画像を指定する」:P.197参照

コラム

表の背景画像の対応状況

Netscape Navigator4.x以降とInternet Explorer3.x以降では、いずれも表の背景画像をサポートしています。

ただし、Internet Explorerでは、表の横列全体である<TR>に対しては、背景画像を指定することができません。

両ブラウザで表示させるためには、表全体に対して指定するか、セルごとに1つ1つ指定する必要があります。

HTML 表にテキストを回り込ませる - 表の位置指定 -

<TABLE align="*位置*">～</TABLE>

位置	left・right・center

Internet Explorer

右の表に示すように、岡蔵さんは現在35歳の男性です。最近の趣味は、ソルトウォーターフィッシングだそうで、会社の同僚と毎週のように海に通っているとか。やはり、原稿を書くよりは魚を釣る方が好きなようで、誘われると締め切りが近くても釣りに行ってしまうそうです。

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

Netscape Navigator

右の表に示すように、岡蔵さんは現在35歳の男性です。最近の趣味は、ソルトウォーターフィッシングだそうで、会社の同僚と毎週のように海に通っているとか。やはり、原稿を書くよりは魚を釣る方が好きなようで、誘われると締め切りが近くても釣りに行ってしまうそうです。

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

解説

表を左右、又は中央に配置します。
左、又は右に配置した場合には、表に続くテキストがその横に回り込みます。回り込みを解除したい場合には、
のclear属性を使用してください。
ただし、これらの属性は廃止予定となっていますので、スタイルシートを使った方がよいでしょう。

Sample

```
<TABLE border="2" align="right">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>
<P>
右の表に示すように、岡蔵さんは現在35歳の男性です。最近の趣味は、ソルトウォーターフィッシングだそうで、会社の同僚と毎週のように海に通っているとか。やはり、原稿を書くよりは魚を釣る方が好きなようで、誘われると締め切りが近くても釣りに行ってしまうそうです。
<BR clear="right">
</P>
```

「テーブル」の「表への回り込みを解除する」:P.109参照
「スタイルシート」の「左右への配置と回り込みを指定する」:P.230参照

表と回り込ませたテキストの間隔を指定する

<TABLE vspace="*縦間隔*" hspace="*横間隔*">～</TABLE>

縦間隔	表の上下の間隔(ピクセル)
横間隔	表の左右の間隔(ピクセル)

Internet Explorer

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

右の表に示すように、岡蔵さんは現在35歳の男性です。最近の趣味は、ソルトウォーターフィッシングだそうで、会社の同僚と毎週のように海に通っているとか。やはり、原稿を書くよりは魚を釣る方が好きなようで、誘われると締め切りが近くても釣りに行ってしまうそうです。

Netscape Navigator

名前	年齢	性別
岡蔵 龍一	35	男
伊藤 清治	29	男

右の表に示すように、岡蔵さんは現在35歳の男性です。最近の趣味は、ソルトウォーターフィッシングだそうで、会社の同僚と毎週のように海に通っているとか。やはり、原稿を書くよりは魚を釣る方が好きなようで、誘われると締め切りが近くても釣りに行ってしまうそうです。

解説

表を左、又は右に配置してテキストをその横に回り込ませた場合の、表との間隔を設定します。
ただし、この属性は一部ブラウザの独自拡張で、HTML4.0では定義されていないものです。表とテキストの間隔(表のマージン)を指定したい場合には、スタイルシートを利用するのが正しいやり方です。

Sample

```
<HTML lang="ja">
<HEAD>
<TITLE>回り込みテキストの間隔サンプル</TITLE>
</HEAD>
<BODY>
```

```
<TABLE border="2" align="left" vspace="10" hspace="30">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>

<P>
右の表に示すように、岡蔵さんは現在35歳の男性です。最近の趣味は、ソルトウォーターフィッシングだそうで、会社の同僚と毎週のように海に通っているとか。やはり、原稿を書くよりは魚を釣る方が好きなようで、誘われると締め切りが近くても釣りに行ってしまうそうです。
<BR clear="left">
</P>

</BODY>
</HTML>
```

「スタイルシート」の「マージンを設定する」:P.205参照

HTML

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

表への回り込みを解除する

<BR clear="*どちら側の表に対して解除するか*">

【どちら側の表に対して解除するか】

left	左側の表に対する回り込みを解除
right	右側の表に対する回り込みを解除
all	両側の表に対する回り込みを解除

Internet Explorer

右の表に示すように、岡蔵さんは現在35歳の男性です。

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

最近の趣味は、ソルトウォーターフィッシングだそうで、会社の同僚と毎週のように海に通っているとか。やはり、原稿を書くよりは魚を釣る方が好きなようで、誘われると締め切りが近くても釣りに行ってしまうそうです。

Netscape Navigator

右の表に示すように、岡蔵さんは現在35歳の男性です。

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

最近の趣味は、ソルトウォーターフィッシングだそうで、会社の同僚と毎週のように海に通っているとか。やはり、原稿を書くよりは魚を釣る方が好きなようで、誘われると締め切りが近くても釣りに行ってしまうそうです。

解説

表を左、又は右に配置してテキストをその横に回り込ませた場合の、回り込みを解除します。
ただし、HTMLで回り込みを制御するための属性は廃止予定となっていますので、スタイルシートを使った方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>表への回り込み解除サンプル</TITLE>
</HEAD>
<BODY>
```

```
<TABLE border="2" align="right">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>

<P>
右の表に示すように、岡蔵さんは現在35歳の男性です。
<BR clear="right">
最近の趣味は、ソルトウォーターフィッシングだそうで、会社の同僚と毎週のように海に通っ
ているとか。
やはり、原稿を書くよりは魚を釣る方が好きなようで、誘われると締め切りが近くても釣り
に行ってしまうそうです。
</P>

</BODY>
</HTML>
```

「テーブル」の「表にテキストを回り込ませる - 表の位置指定 -」:P.106参照
「スタイルシート」の「回り込みを解除する」:P.231参照

セルを連結する

```
<TH rowspan="縦方向の連結数">～</TH>
<TH colspan="横方向の連結数">～</TH>
<TD rowspan="縦方向の連結数">～</TD>
<TD colspan="横方向の連結数">～</TD>
```

縦方向の連結数	そのセルから下方向へ連結するセルの数
横方向の連結数	そのセルから右方向へ連結するセルの数

Internet Explorer

会員		地域
岡蔵　龍一	男	札幌
伊藤　清治	男	
山田　花子	女	

Netscape Navigator

会員		地域
岡蔵　龍一	男	札幌
伊藤　清治	男	
山田　花子	女	

解説

この属性を利用すると、指定した数のセルを1つにしたセルを作成することができます。

rowspan属性はそのセルから下方向に、colspan属性はそのセルから右方向に、複数のセル分の領域を持った1つのセルを作成します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>セルの連結サンプル</TITLE>
</HEAD>
<BODY>
<TABLE border="2">
<TR>
<TH colspan="2">会員</TH>
<TH>地域</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD>
<TD>男</TD>
<TD rowspan="3">札幌</TD>
</TR>
```

```
<TR>
<TD>伊藤　清治</TD>
<TD>男</TD>
</TR>
<TR>
<TD>山田　花子</TD>
<TD>女</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

コラム

セルの連結に関するHTML4.01での仕様変更

HTML4.01では、セルの連結に関して、一部の仕様が変更になりました。

変更されたのは、本書の解説では特に触れていないのですが、rowspan属性とcolspan属性の値として「0」を指定した場合の表示方法に関する部分です。具体的にどう変更されたのかというと、HTML4.0では値として「0」が指定されると、そのセル以降のすべてのセルが連結されることになっていたのですが、HTML4.01では、それがグループ内(THEAD・TBODY・TFOOT・COLGROUP)のすべてのセルというようにグループの範囲に限定されました。

いずれにしても、この指定方法は現在では利用できませんので、あくまで知識として持っていてください。

HTML

セルの大きさを指定する

```
<TH width="幅" height="高さ">～</TH>
<TD width="幅" height="高さ">～</TD>
```

幅	ピクセル
高さ	ピクセル

Internet Explorer

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

Netscape Navigator

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

解説

width属性はセルの幅を、height属性はセルの高さをピクセル単位で設定します。ただし、これらの属性は廃止予定となっていますので、スタイルシートを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>セルの大きさ指定サンプル</TITLE>
</HEAD>
<BODY>
<TABLE border="2">
<TR>
<TH width="120" height="50">名前</TH>
<TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

「スタイルシート」の「幅と高さを指定する」:P.219参照

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

HTML

セル内のテキストの位置を指定する

HTML

```
<TR align="横位置" valign="縦位置">～</TR>
<TH align="横位置" valign="縦位置">～</TH>
<TD align="横位置" valign="縦位置">～</TD>
<THEAD align="横位置" valign="縦位置">～</THEAD>
<TBODY align="横位置" valign="縦位置">～</TBODY>
<TFOOT align="横位置" valign="縦位置">～</TFOOT>
<COL align="横位置" valign="縦位置">～</COL>
<COLGROUP align="横位置" valign="縦位置">～</COLGROUP>
```

【横位置】

left	左揃え(<TD>のデフォルト)
right	右揃え
center	中央揃え(<TH>のデフォルト)
justify	両端揃え
char	特定の文字の位置(小数点など)

【縦位置】

top	上
middle	中央(デフォルト)
bottom	下
baseline	横方向の列での1行目のベースライン

Internet Explorer

	align="left"	align="center"	align="right"
valign="top"	位置A	位置B	位置C
valign="middle"	位置D	位置E	位置F
valign="bottom"	位置G	位置H	位置I

Netscape Navigator

	align="left"	align="center"	align="right"
valign="top"	位置A	位置B	位置C
valign="middle"	位置D	位置E	位置F
valign="bottom"	位置G	位置H	位置I

解説

align属性はセル内での横方向の表示位置を、valign属性は縦方向の表示位置を設定します。

HTML4.0からは、align属性の値として「char」が採用され、小数点などの位置を揃えることができる仕様となりました。この場合、char属性で位置を揃える文字(デフォルトは小数点)を指定し、charoff属性(値はピクセルまたは%)で位置を指定します。しかし、「justify」と同様、現在は、ほとんどのブラウザでサポートされていないようです。

表を作る

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>セル内のテキストの位置揃えサンプル</TITLE>
</HEAD>
<BODY>
<TABLE border="2">
<TR>
<TH></TH>
<TH>align=&quot;left&quot;</TH>
<TH>align=&quot;center&quot;</TH>
<TH>align=&quot;right&quot;</TH>
</TR>
<TR valign="top">
<TH height="80">valign=&quot;top&quot;</TH>
<TD align="left">位置A</TD>
<TD align="center">位置B</TD>
<TD align="right">位置C</TD>
</TR>
<TR valign="middle">
<TH height="80">valign=&quot;middle&quot;</TH>
<TD align="left">位置D</TD>
<TD align="center">位置E</TD>
<TD align="right">位置F</TD>
</TR>
<TR valign="bottom">
<TH height="80">valign=&quot;bottom&quot;</TH>
<TD align="left">位置G</TD>
<TD align="center">位置H</TD>
<TD align="right">位置I</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

「テーブル」の「横列をグループ化する」:P.118参照
「テーブル」の「縦列をグループ化する」:P.120参照
「テーブル」の「縦列に幅や行揃えをまとめて指定する」:P.122参照

NN2.0
NN3.0
NN4.0

IE3.0
IE4.0
IE5.0

HTML

セル内での改行を禁止する

```
<TH nowrap>~</TH>
<TD nowrap>~</TD>
```

Internet Explorer

通常の状態

名前	住所
岡蔵 龍一	〒060-1234 北海道札幌市中央区大通り西10丁目 20−30 第5ブルービル21F

nowrap属性で自動的な改行を禁止した状態

名前	住所
岡蔵 龍一	〒060-1234 北海道札幌市中央区大通り西10丁目20−

通常の状態

名前	住所
岡蔵 龍一	〒060-1234 北海道札幌市中央区大通り西10丁目20−30 第5ブルービル21F

nowrap属性で自動的な改行を禁止した状態

名前	住所
岡蔵 龍一	〒060-1234 北海道札幌市中央区大通り西10丁目20−30 第5ブルービル21F

Netscape Navigator

通常の状態

名前	住所
岡蔵 龍一	〒060-1234 北海道札幌市中央区大通り西10丁目 20−30 第5ブルービル21F

nowrap属性で自動的な改行を禁止した状態

名前	住所
岡蔵 龍一	〒060-1234 北海道札幌市中央区大通り西10丁目20−

通常の状態

名前	住所
岡蔵 龍一	〒060-1234 北海道札幌市中央区大通り西10丁目20−30 第5ブルービル21F

nowrap属性で自動的な改行を禁止した状態

名前	住所
岡蔵 龍一	〒060-1234 北海道札幌市中央区大通り西10丁目20−30 第5ブルービル21F

解説

通常、セルの表示幅はウインドウの幅などに合わせて自動的に調整されるため、セル内のテキストが長い場合には、自動的に改行されてしまいます。この自動的な改行を禁止するのがnowrap属性です。
この属性を指定すると、意図的に改行させない限り自動的な改行は行われなくなります。
この属性は廃止予定となっていますので、本来はスタイルシート(white-space: no wrap)を利用した方がよいのですが、現在のところほとんどのブラウザでサポートされていないようです。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>セル内での改行の禁止サンプル</TITLE>
</HEAD>
<BODY>

<P>通常の状態</P>
<TABLE border="2">
<TR>
<TH>名前</TH>
<TH>住所</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD>
<TD>〒060-1234　北海道札幌市中央区大通り西10丁目20−30　第5ブルービル21F
</TD>
</TR>
</TABLE>

<P>nowrap属性で自動的な改行を禁止した状態</P>
<TABLE border="2">
<TR>
<TH>名前</TH>
<TH>住所</TH>
</TR>
<TR>
<TD nowrap>岡蔵　龍一</TD>
<TD nowrap>〒060-1234　北海道札幌市中央区大通り西10丁目20−30　第5ブルー
ビル21F</TD>
</TR>
</TABLE>

</BODY>
</HTML>
```

NN2.0 NN3.0 NN4.0 IE3.0 IE4.0 IE5.0

HTML

横列をグループ化する

IE3.0 IE4.0 IE5.0

<THEAD>～</THEAD> ←表のヘッダ部分
<TFOOT>～</TFOOT> ←表のフッタ部分
<TBODY>～</TBODY> ←表の本体(データ)部分

Internet Explorer

個人情報		
名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

Netscape Navigator

個人情報		
名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

解説

表の横方向の列を、ヘッダ・本体・フッタの各部分として、まとめてグループ化する場合に利用します。
これによって、その範囲全体に対して属性やスタイルシートを指定できるようになるばかりでなく、将来ブラウザが対応すれば、ヘッダとフッタを固定した状態で本体部分をスクロールしたり、印刷時に各ページにヘッダとフッタを印刷することなどが可能になります。
これらの要素は、表の本体(データ)部分が完全にロードされる前にフッタを表示できるように、必ずTHEAD要素・TFOOT要素・TBODY要素の順になるように記述してください。TBODY要素は、必要に応じて複数配置することができます。
このサンプルでは、ヘッダ部分と本体部分に対して、class属性を使用してスタイルシートを適用させています。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>横列のグループ化サンプル</TITLE>
<STYLE type="text/css">
<!--
.tableheader {
    color: #FFFFFF;
    background-color: #006666
}
```

表を作る

```
.tabledata {
    background-color: #FFFFFF;
}
-->
</STYLE>
</HEAD>
<BODY>
<TABLE border="2">
<THEAD class="tableheader">
<TR>
<TH colspan="3">個人情報</TH>
</TR>
<TR>
<TH>名前</TH><TH>年齢</TH><TH>性別</TH>
</TR>
</THEAD>
<TBODY class="tabledata">
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TBODY>
</TABLE>
</BODY>
</HTML>
```

「テーブル」の「表内のセルを区切る線の表示形式を指定する」:P.100参照
「テーブル」の「セル内のテキストの位置を指定する」:P.114参照

HTML

縦列をグループ化する

IE3.0
IE4.0
IE5.0

<COLGROUP span="*縦列数*">～</COLGROUP>
<COLGROUP span="*縦列数*" width="*幅*">～</COLGROUP>

縦列数	グループ化する縦列の数(省略した場合は1)
幅	ピクセル・%・*

Internet Explorer

20代	8	401	254
30代	85	612	348
40代	55	312	288

Netscape Navigator

20代	8	401	254
30代	85	612	348
40代	55	312	288

解説

表の縦列を(構造的な意味で)まとめてグループ化する場合に利用します。
これによって、複数の縦列に対して、まとめて幅や行揃えなどの属性やスタイルシートを指定できるようになります。設定したグループ内で、更に縦列に対して個別の指定をしたい場合には、次に説明するCOL要素を使用します。この場合は、span属性はCOLGROUP要素で指定せずに、COL要素側で指定するようにしてください。
COLGROUP要素は、配置する位置に注意が必要です。TABLE要素とそれに続くCAPTION要素があればその直後で、THEAD要素(なければTR要素)よりも前の位置に配置するようにしてください。また、この要素の内容として配置できるのは、COL要素だけです。終了タグ</COLGROUP>は、省略することも可能です。
このサンプルでは、最初の縦列に対して、スタイルシートで色を指定しています。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>縦列のグループ化サンプル</TITLE>
<STYLE TYPE="text/css">
<!--
.tableheader {
    background-color: #006666;
    color: #FFFFFF;
}
-->
</STYLE>
</HEAD>
```

```
<BODY>
<TABLE border="2">
<COLGROUP class="tableheader">
<COLGROUP span="3" width="60" align="right">
<TR>
<TH>20代</TH><TD>8</TD><TD>401</TD><TD>254</TD>
</TR>
<TR>
<TH>30代</TH><TD>85</TD><TD>612</TD><TD>348</TD>
</TR>
<TR>
<TH>40代</TH><TD>55</TD><TD>312</TD><TD>288</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

「テーブル」の「表内のセルを区切る線の表示形式を指定する」:P.100参照
「テーブル」の「セル内のテキストの位置を指定する」:P.114参照
Tip「『*』による幅や高さの割合の指定」:P.170参照

縦列に幅や行揃えをまとめて指定する

```
<COL span="縦列数">
<COL span="縦列数" width="幅">
```

縦列数	属性をまとめて指定する縦列の数(省略した場合は1)
幅	ピクセル・%・*

Internet Explorer

名前	年齢	体重	性別
岡蔵　龍一	35	70	男
伊藤　清治	29	64	男
山田　花子	22	45	女

Netscape Navigator

名前	年齢	体重	性別
岡蔵　龍一	35	70	男
伊藤　清治	29	64	男
山田　花子	22	45	女

解説

表の縦列を構造的にグループ化するのではなく、幅や行揃えなどの属性やスタイルシートをまとめて指定したい場合に利用します。COL要素は、配置する位置に注意が必要です。TABLE要素とそれに続くCAPTION要素の直後で、THEAD要素(なければTR要素)よりも前の位置に配置するようにしてください。

Sample

```
<TABLE border="2">
<COL width="120">
<COL span="2" align="right">
<COL align="center">
<TR>
<TH>名前</TH><TH>年齢</TH><TH>体重</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>70</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>64</TD><TD>男</TD>
</TR>
<TR>
<TD>山田　花子</TD><TD>22</TD><TD>45</TD><TD>女</TD>
</TR>
</TABLE>
```

「テーブル」の「表内のセルを区切る線の表示形式を指定する」:P.100参照
「テーブル」の「セル内のテキストの位置を指定する」:P.114参照
Tip「『*』による幅や高さの割合の指定」:P.170参照

HTML

画像を配置する

<IMG src="*URL*" width="*幅*" height="*高さ*" alt="*代替文字*">

URL	画像ファイルのURL
幅	画像の幅(ピクセルまたは%)
高さ	画像の高さ(ピクセルまたは%)
代替文字	画像の代わりとなるテキスト

Internet Explorer

Netscape Navigator

解説

HTML文書内に画像を配置する場合に使用します。
画像の形式としては、一般にGIF・JPEG・PNG形式が利用できます。
width属性とheight属性は、それぞれ画像を実際に表示させる幅と高さを指定します。画像の実サイズ以外の値を指定することも可能です。これらを指定しておくことで画像を表示させるスペースが確保できるので、画像データが完全にロードされる前に、その後の内容も表示できるようになります。alt属性には、画像が表示できない場合に画像のかわりに表示させるテキストを指定します。この属性は「画像の説明」ではなく、あくまで「画像の代わりに表示させるテキスト」ですので、前後の文脈をよく考えて、意味が通るものを指定してください。
なお、HTML4.0では、src属性とalt属性は必ず指定する決まりになっています。

Sample

```
<BODY bgcolor="white">
<P>
<IMG src="../images/fish.gif" width="230" height="145"
 alt="サンプル画像">
</P>
</BODY>
```

「アクセシビリティ」の「画像の代わりのテキストを指定しておく」:P.238参照

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

HTML

画像の枠線を設定する

```
<IMG src="URL" alt="代替文字" border="枠の太さ">
```

枠の太さ　ピクセル

Internet Explorer

リンクしている場合

※左から、「border="0"」「border属性なし」「border="5"」

リンクしていない場合

※左から、「border属性なし」「border="2"」「border="5"」

Netscape Navigator

リンクしている場合

※左から、「border="0"」「border属性なし」「border="5"」

リンクしていない場合

※左から、「border属性なし」「border="2"」「border="5"」

解説

画像の回りに表示される枠の太さを設定します。
一般的なブラウザでは、通常は枠が表示されていませんが、この属性を指定することで枠が表示されるようになります。また、画像をリンクさせるようにすると、それを示すために枠が自動的に表示されますが、この枠の太さも同様に設定できます。
この場合、枠の太さに0を指定することで枠を消すこともできます。
ただし、この属性は廃止される予定となっていますので、枠の太さを設定する場合には、スタイルシートを利用したほうがよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>画像の枠線サンプル</TITLE>
</HEAD>
<BODY bgcolor="white">
```

```
<H1>リンクしている場合</H1>
<P>
※左から、「border="0"」「border属性なし」「border="5"」<BR>
<A href="next.html">
<IMG src="../images/leaf.gif" width="140" height="100"
 alt="" border="0"></A>
<A href="next.html">
<IMG src="../images/leaf.gif" width="140" height="100"
 alt=""></A>
<A href="next.html">
<IMG src="../images/leaf.gif" width="140" height="100"
 alt="" border="5"></A>
</P>

<H1>リンクしていない場合</H1>
<P>
※左から、「border属性なし」「border="2"」「border="5"」<BR>
<IMG src="../images/leaf.gif" width="140" height="100"
alt="">
<IMG src="../images/leaf.gif" width="140" height="100"
 alt="" border="2">
<IMG src="../images/leaf.gif" width="140" height="100"
 alt="" border="5">
</P>

</BODY>
</HTML>
```

「スタイルシート」の「枠の太さを指定する」:P.220参照

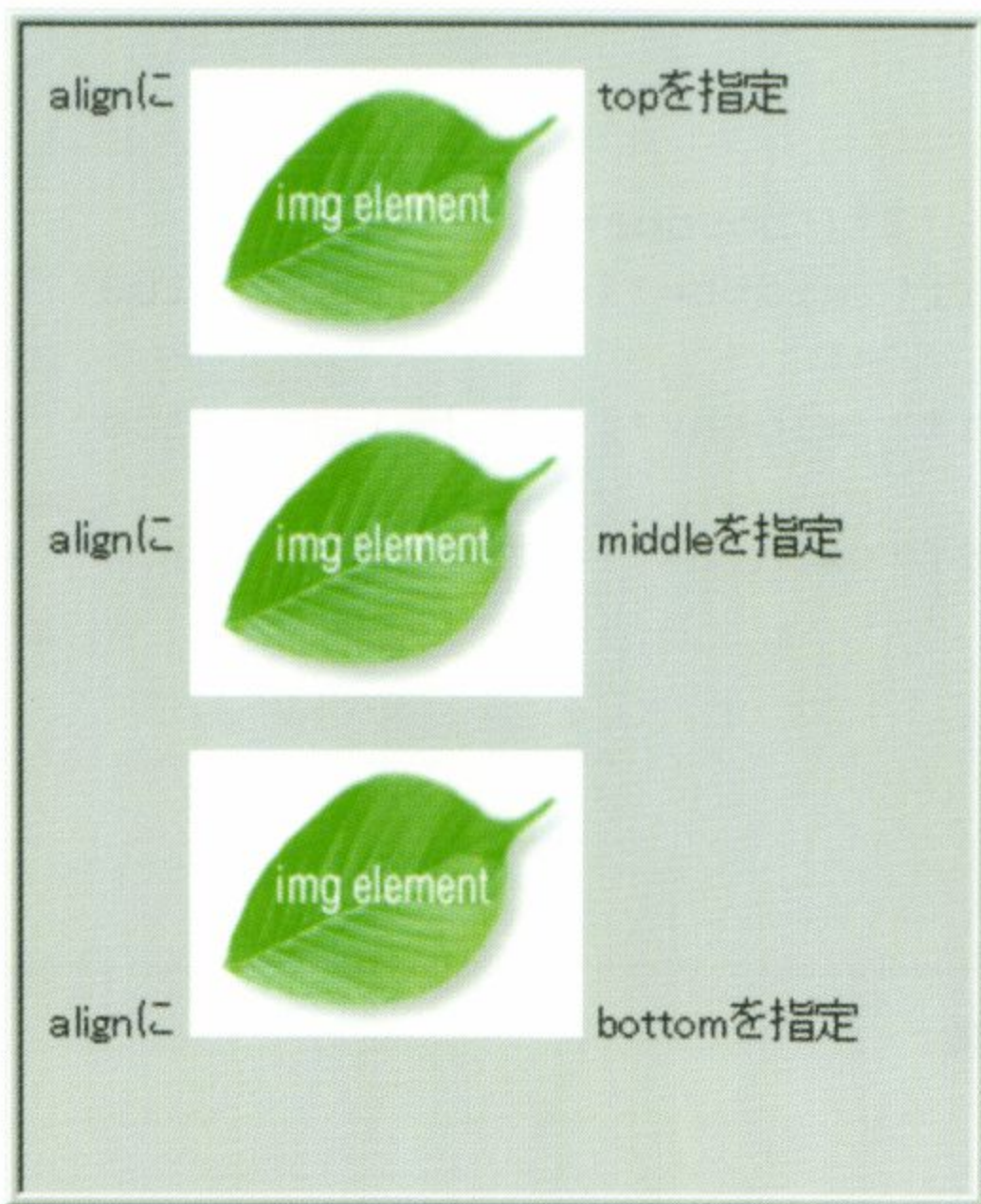
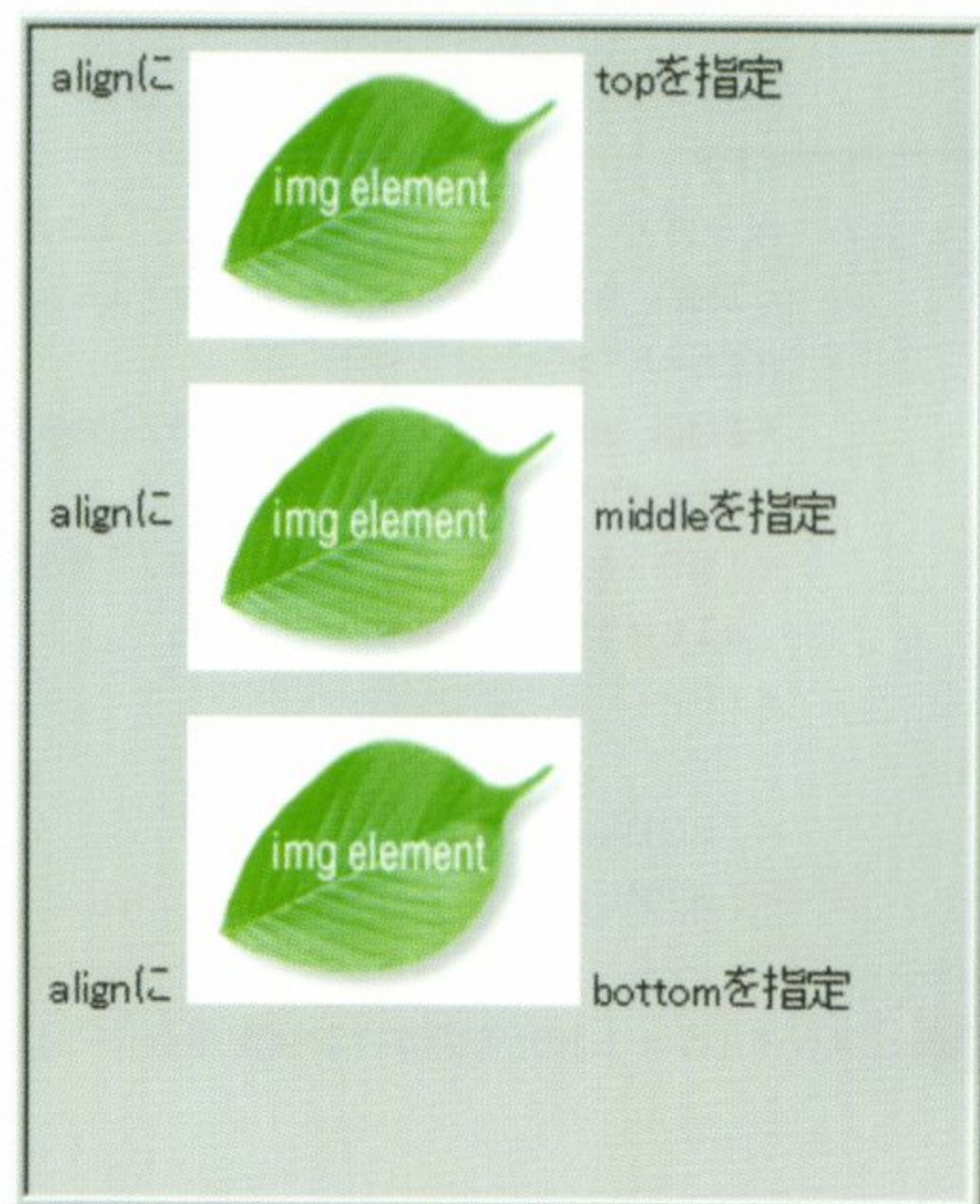
HTML

画像とテキストの縦の位置関係を指定する

<IMG src="*URL*" alt="*代替文字*" align="*位置*">

【位置】

top	画像の上とテキストの上を揃える
middle	画像の中心とテキストのベースラインを揃える
bottom	画像の下とテキストのベースラインを揃える(デフォルト)

Internet Explorer

Netscape Navigator

解説

画像(IMG要素)は、テキストの行の中に配置することができるインライン要素です。align属性は、1つの行の中に画像とテキストが含まれる場合の、画像とテキストの縦の位置関係を設定します。ただし、この属性は廃止予定となっていますので、スタイルシートを使った方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>画像とテキストの縦の位置関係サンプル</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
```

```
<P>alignに
<IMG src="../images/leaf.gif" width="140" height="100"
 alt="" align="top">
topを指定
</P>

<P>alignに
<IMG src="../images/leaf.gif" width="140" height="100"
 alt="" align="middle">
middleを指定
</P>

<P>alignに
<IMG src="../images/leaf.gif" width="140" height="100"
 alt="" align="bottom">
bottomを指定
</P>

</BODY>
</HTML>
```

「スタイルシート」の「縦方向の位置関係を指定する」:P.208参照

HTML

画像にテキストを回り込ませる - 画像の位置指定 -

```
<IMG src="URL" alt="代替文字" align="位置">
```

位置	left · right

Internet Explorer

Netscape Navigator

IMG要素に対して「align="left"」を指定すると、このように画像が左側に配置されて、それに続くテキストが右側に表示されます。右側に収まりきらないテキストは、画像の下に表示されます。この回り込みを解除するためには、BR要素のclear属性を使用します。また、vspace属性やhspace属性を使用すると、画像とテキストとの間隔を設定することができます。

解説

画像を左、又は右に配置して、その反対側にテキストを回り込ませます。回り込みを解除したい場合には、
のclear属性を使用してください。ただし、これらの属性は廃止予定となっていますので、スタイルシートを使った方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>テキストの回り込みサンプル</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<P>
<IMG src="../images/fish.gif" width="230" height="145"
 alt="サンプル画像" align="left">
IMG 要素に対して「align="left"」を（中略）設定することができます。
<BR clear="left">
</P>
</BODY>
</HTML>
```

「画像とマルチメディア」の「画像への回り込みを解除する」：P.130参照
「スタイルシート」の「左右への配置と回り込みを指定する」：P.230参照

HTML 画像と回り込ませたテキストの間隔を指定する

<IMG src="*URL*" alt="*代替文字*" vspace="*縦間隔*" hspace="*横間隔*">

縦間隔	画像の上下の間隔(ピクセル)
横間隔	画像の左右の間隔(ピクセル)

Internet Explorer

image

IMG要素に対して「align="left"」を指定すると、このように画像が左側に配置されて、それに続くテキストが右側に表示されます。右側に収まりきらないテキストは、画像の下に表示されます。vspace属性で、この下に表示されたテキストと画像との間隔を指定できますが、これはそのテキストの行の高さによって微妙に違ってきます。

Netscape Navigator

image

IMG要素に対して「align="left"」を指定すると、このように画像が左側に配置されて、それに続くテキストが右側に表示されます。右側に収まりきらないテキストは、画像の下に表示されます。vspace属性で、この下に表示されたテキストと画像との間隔を指定できますが、これはそのテキストの行の高さによって微妙に違ってきます。

解説

画像を左、又は右に配置してテキストをその横に回り込ませた場合の、画像との間隔を設定します。
ただし、これらの属性は廃止予定となっていますので、スタイルシートを使った方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>回り込ませたテキストとの間隔サンプル</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<P>
<IMG src="../images/fish.gif" width="230" height="145"
 alt="サンプル画像" align="left" vspace="5" hspace="30">
IMG 要素に対して「align="left"」を（中略）高さによって微妙に違ってきます。
<BR clear="left">
</P>
</BODY>
</HTML>
```

「スタイルシート」の「マージンを設定する」:P.205参照

HTML

画像への回り込みを解除する

<BR clear="*どちら側の画像に対して解除するか*">

【どちら側の画像に対して解除するか】

left	左側の画像に対する回り込みを解除
right	右側の画像に対する回り込みを解除
all	両側の画像に対する回り込みを解除

Internet Explorer

IMG要素に対して「align="left"」を指定すると、このように画像が左側に配置されて、それに続くテキストが右側に表示されます。

この回り込みを解除するためには、BR要素のclear属性を使用します。

Netscape Navigator

IMG要素に対して「align="left"」を指定すると、このように画像が左側に配置されて、それに続くテキストが右側に表示されます。

この回り込みを解除するためには、BR要素のclear属性を使用します。

解説

画像を左、又は右に配置してテキストをその横に回り込ませた場合の、回り込みを解除します。
ただし、HTMLで回り込みを制御するための属性は廃止予定となっていますので、スタイルシートを使った方がよいでしょう。

Sample

```
<P>
<IMG src="../images/fish.gif" width="230" height="145"
 alt="サンプル画像" align="left">
IMG要素に対して「align="left"」を指定すると、このように画像が左側に配置されて、
それに続くテキストが右側に表示されます。
<BR clear="left">
</P>

<P>
この回り込みを解除するためには、BR要素のclear属性を使用します。
</P>
```

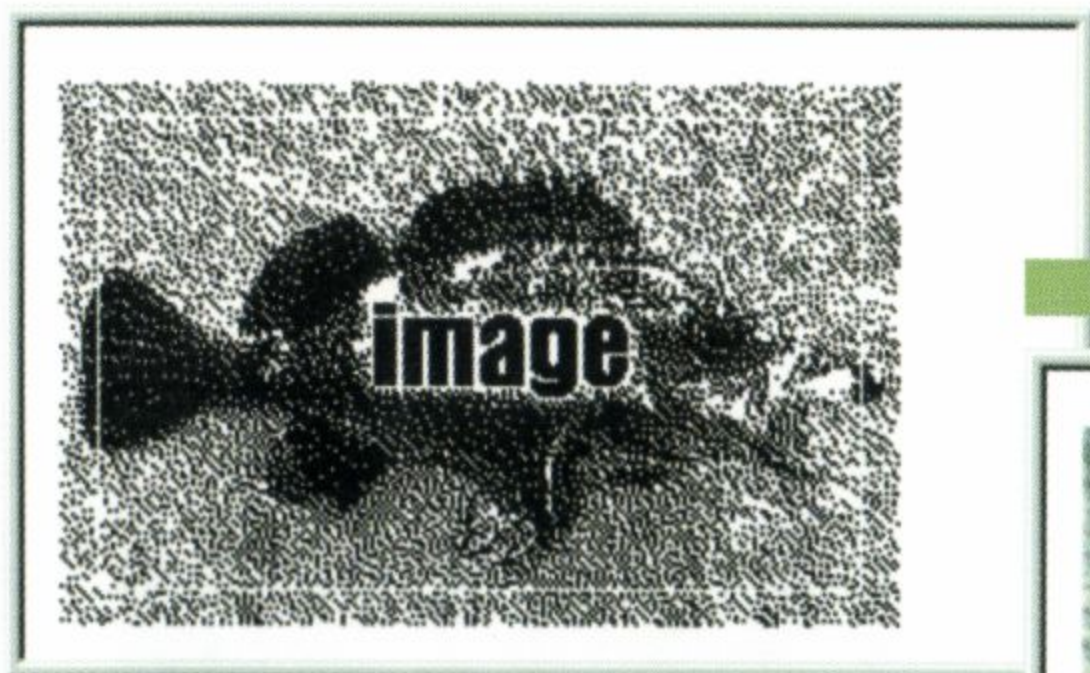
「画像とマルチメディア」の「画像にテキストを回り込ませる - 画像の位置指定 -」:P.128参照
「スタイルシート」の「回り込みを解除する」:P.231参照

画像を２段階で表示させる

NN2.0 NN3.0 NN4.0 IE4.0 IE5.0

<IMG src="*URL1*" lowsrc="*URL2*" alt="*代替文字*">

URL1	最終的に表示される画像ファイルのURL
URL2	最初に表示させるデータ量の少ない画像ファイルのURL

Netscape Navigator

After

解説

データ量が多く、ロードに時間がかかりそうな画像を使用したい場合に、最初に仮のデータ量の少ない画像を表示させておくことができます。
この場合、データ量の多い画像のロードと共に、画像が徐々に置き換わっていきます。画像データは、一旦読み込まれると通常はキャッシュされるため、次に表示させた時にはデータ量の少ない画像は表示されなくなります。
この属性は一部ブラウザの独自拡張で、HTML4.0では定義されていないものですので注意してください。

Sample

```
<HTML lang="ja">
<HEAD>
<TITLE>画像を2段階で表示させるサンプル</TITLE>
</HEAD>
<BODY bgcolor="white">
<P>
<IMG src="../images/fish.gif" lowsrc="../images/bw.gif"
 width="230" height="145" alt="サンプル画像">
</P>
</BODY>
</HTML>
```

HTML

イメージマップを作成する

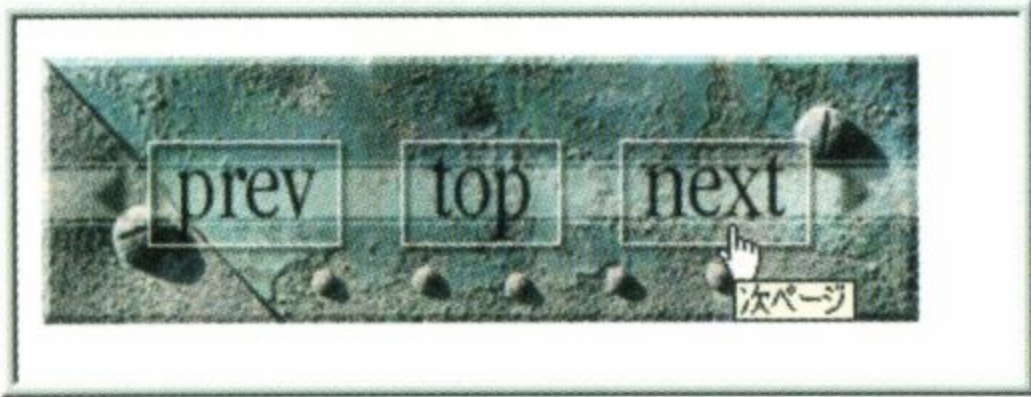
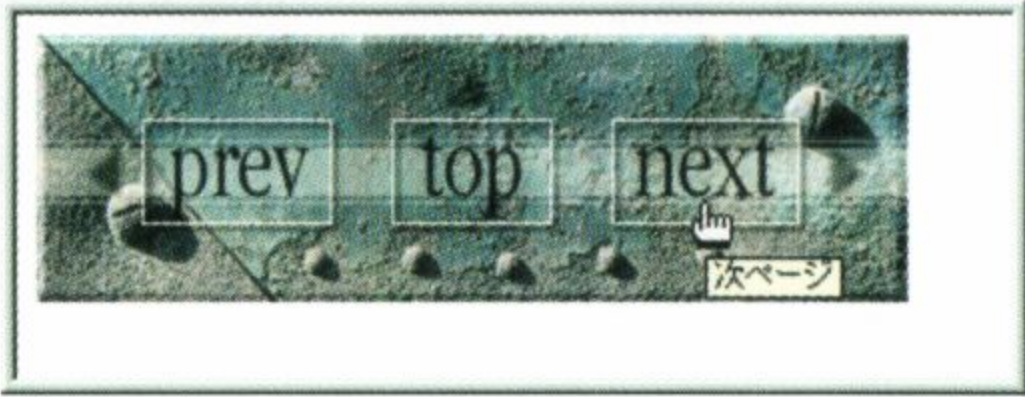
```
<IMG src="URL" alt="代替文字" usemap="#マップ名">
<MAP name="マップ名">～</MAP>
<AREA shape="形状" coords="座標" href="URL" alt="代替文字">
```

マップ名	イメージマップの名前	
形状	rect	四角形
	circle	円
	poly	多角形
	default	全体
座標	形状別の座標値	

Internet Explorer　　Netscape Navigator

prev　top　next　前ページ

prev　top　next　トップページ

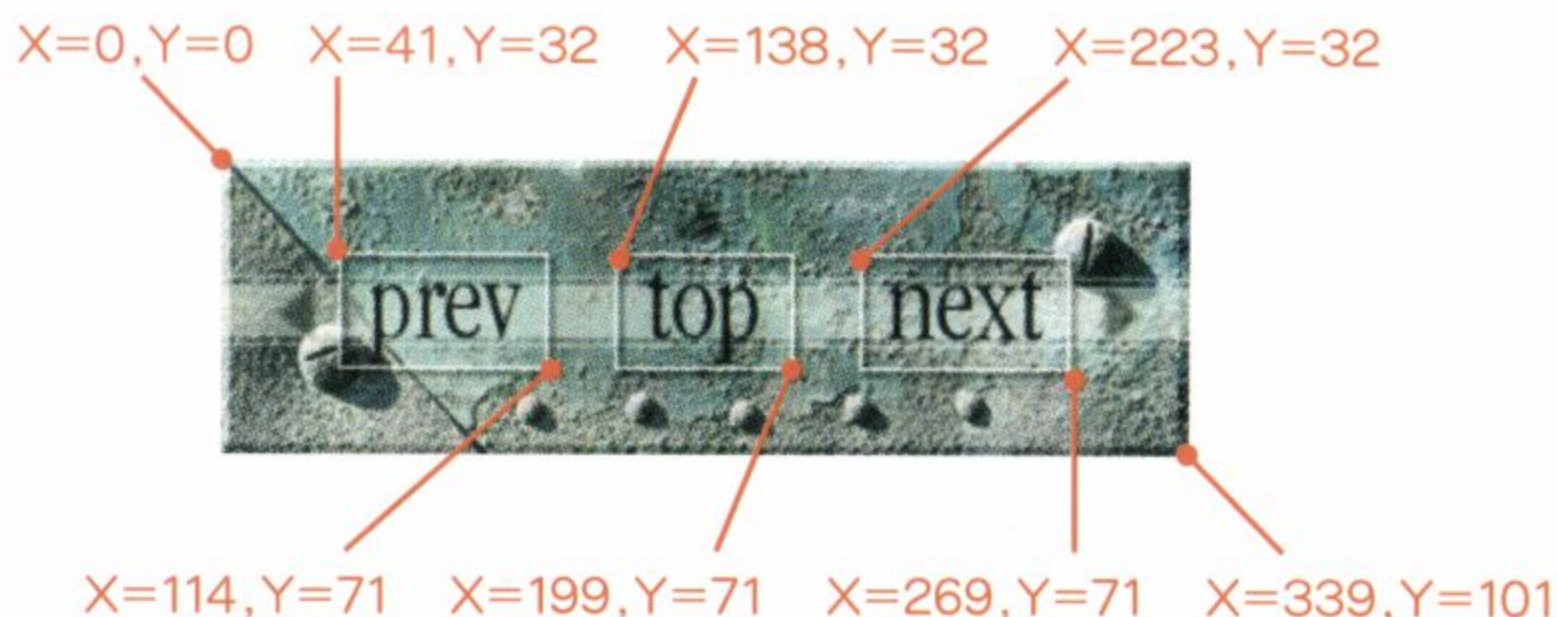
prev　top　next　次ページ

解説

イメージマップとは、画像の特定の領域をマウス・クリックに反応するようにして、他の文書へリンクするようにしたものです。

MAP要素は、その内容がイメージマップの定義であることを示し、それに名前を付けます。MAP要素内で実際にクリックに反応する領域と、そのリンク先を指定するのがAREA要素です。後は、IMG要素のusemap属性で、定義したイメージマップの名前を頭に「#」を付けて指定すれば、イメージマップの完成です。

AREA要素のcoords属性で指定する座標の指定方法は、shape属性で指定する領域の形状によって異なります。指定方法は次の通りで、各座標値はピクセル単位で「,」で区切って指定します。

coords属性での座標の指定方法

四角形(rect)の場合	「左上のX座標」・「左上のY座標」・「右下のX座標」・「右下のY座標」
円(circle)の場合	「中心のX座標」「中心のY座標」「半径」
多角形(poly)の場合	すべての角の座標を「X座標」・「Y座標」の順で指定

X=0,Y=0　X=41,Y=32　X=138,Y=32　X=223,Y=32

prev　top　next

X=114,Y=71　X=199,Y=71　X=269,Y=71　X=339,Y=101

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>イメージマップ・サンプル</TITLE>
</HEAD>
<BODY bgcolor="white">
<P>
<IMG src="../images/imap.jpg" width="339" height="101"
 alt="イメージマップ・サンプル画像" border="0" usemap="#map1">
<MAP name="map1">
<AREA shape="rect" coords="41,32,114,71" href="prev.html"
 alt="前ページ">
<AREA shape="rect" coords="138,32,199,71" href="top.html"
 alt="トップページ">
<AREA shape="rect" coords="223,32,296,71" href="next.
html" alt="次ページ">
</MAP>
</P>
</BODY>
</HTML>
```

「アクセシビリティ」の「画像の代わりのテキストを指定しておく」:P.238参照

NN4.0 IE3.0 IE4.0 IE5.0

HTML

様々な形式のデータを配置する

<OBJECT data="*URL*" type="*MIMEタイプ*" ・・・>～</OBJECT>

URL	配置するデータのURL
MIMEタイプ	配置するデータのMIMEタイプ

解説

様々な形式のデータを配置することのできる汎用的な要素です。
具体的には、画像・動画・JAVAアプレット・プラグインデータ・他のHTML文書などを配置することができます。この要素は、現在データの形式ごとに使われている、IMG要素・APPLET要素・EMBED要素・BGSOUND要素などの要素を統一して扱えるようにする目的で作成されたものです。この要素を使用すると、ブラウザがここで指定した形式のデータを取り扱うことができる場合は、<OBJECT>～</OBJECT>の間の内容は無視されます(ただし、PARAM要素やMAP要素は除く)。したがって、要素内容には、指定した形式のデータをブラウザが取り扱うことができない場合に、表示させたい内容を入れておきます。
これを利用して、利用して欲しいデータ形式から順にOBJECT要素を入れ子にしておくと、ブラウザが利用可能な最も外側のデータ形式が採用されることになります。ただし、現在のところ、この要素を正しく扱えるブラウザはほとんどありません。

※この要素は多くの種類のデータ形式をサポートしているため、実際にはかなりの数の属性がありますが、ここでは省略しています。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>データ形式・サンプル</TITLE>
</HEAD>
<BODY bgcolor="white">
<P>
<OBJECT data="fireworks.mpeg" type="application/mpeg">
<OBJECT data="fireworks.gif" type="image/gif">
色鮮やかな花火が印象的だった。
</OBJECT>
</OBJECT>
</P>
</BODY>
</HTML>
```

画像を配置する

HTML

プラグインを利用するデータを配置する

```
<EMBED src="URL" width="幅" height="高さ">
<NOEMBED>～</NOEMBED>
```

URL	プラグインが使用するデータのURL
幅	ピクセル
高さ	ピクセル

Internet Explorer

ヒラツメガニ捕獲シミュレーション

START

SCORE = 0

SCORE = 0

解説 EMBED要素は、プラグイン機能を利用する場合に使用する要素です。
この要素は、OBJECT要素が定義される以前に一部のブラウザが独自に採用したもので、HTML4.0では定義されていないものです。しかし、まだOBJECT要素が完全には動作しないブラウザが多いため、現在でも利用されています。

基本的には前ページの書式で紹介した属性が利用できますが、使用するプラグインによって他の様々な属性が利用可能です。詳しくは、各プラグインのマニュアルなどを参照してください。NOEMBED要素には、プラグインが利用できない場合に表示させたい内容を入れておきます。ここで指定した内容は、プラグインが利用可能な場合には表示されません。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>プラグイン・サンプル</TITLE>
</HEAD>
<BODY bgcolor="white">
<P align="center">
<EMBED src="crab98.dcr" width="464" height="372">
<NOEMBED>
このシミュレーションを実行するためには、
<A href="http://www.macromedia.com/">Shockwaveプラグイン</A>
が必要です。
</NOEMBED>
</P>
</BODY>
</HTML>
```

JAVAアプレットを配置する

<APPLET code="クラスファイル名" width="幅" height="高さ">〜</APPLET>
<PARAM name="パラメータ名" value="パラメータの値">〜</PARAM>

幅	ピクセルまたは%
高さ	ピクセルまたは%

解説

APPLET要素は、JAVAアプレットを利用する場合に使用する要素です。
実際には、上の書式で紹介した属性以外にも利用可能な属性がありますが、ここでは省略します。<APPLET>〜</APPLET>の範囲には、JAVAアプレットが利用できない場合に表示させたい内容を入れておきます。ここで指定した内容は、JAVAアプレットが動作可能な場合には表示されません。
PARAM要素を利用すると、JAVAアプレット実行時に必要な値を指定しておくことができます。この場合、PARAM要素は<APPLET>〜</APPLET>の範囲の最初に記述してください。
APPLET要素は廃止される予定となっており、かわりにOBJECT要素を使うことが推奨されています。しかし、まだOBJECT要素が完全には動作しないブラウザが多いため、現在でも利用されています。
JAVAアプレットについての詳細は、専門書を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>JAVAアプレット・サンプル</TITLE>
</HEAD>
<BODY bgcolor="white">
<P>
<APPLET code="colorchart.class" width="475" height="400">
JAVAが利用できない環境の方は、<A href="colorchart2.html">
カラーチャート2</A>をご利用ください。
</APPLET>
</P>
</BODY>
</HTML>
```

入力フォームを作る

HTML

<FORM action="*URL*" method="*HTTPメソッド*" enctype="*MIMEタイプ*" target="*ウインドウ名*">～</FORM>

URL	フォームを送信して処理するプログラムのURL
HTTPメソッド	データの送信方法(get・post)
MIMEタイプ	内容を送信する際のMIMEタイプ
ウインドウ名	送信した結果を表示するウインドウまたはフレーム名

Internet Explorer

お名前: shuwa taro
メール: taro@shuwa.ne.jp
男性 女性
送信 クリア

Netscape Navigator

お名前: shuwa hanako
メール: hana@shuwa.ne.jp
男性 女性
送信 クリア

解説

指定した範囲が、ユーザーが入力して送信可能なフォームであることを示します。<FORM>～</FORM>の範囲には、ユーザーが入力や選択などの操作をするための要素(INPUT要素・BUTTON要素・SELECT要素・TEXTAREA要素)を配置します。ユーザーが送信ボタン(INPUT要素かBUTTON要素で「type="submit"」が指定されているもの)を押した段階で、このフォームに記入された内容は、action属性で指定されたURLのCGIプログラムに送信されます。

HTTPメソッドは、データの送信方法を設定します。「get」を指定すると、action属性で指定されているURLにデータを追加した状態で送信されます。この方法は、サーチエンジンなどのフォームでよく利用されています。「post」を指定すると、データ部分だけが送信されます。一般に、送信する内容が日本語で、データ量が多い場合には、こちらを使用します。

enctype属性で指定するMIMEタイプは、デフォルトでは「application/x-www-form-urlencoded」になっています。通常はこれを利用してCGIで処理を行いますが、INPUT要素の「type="file"」でファイルを送信する場合には、MIMEタイプに「multipart/form-data」を設定し、HTTPメソッドには「post」を指定します。

HTML4.0では、action属性で示す送信先は必ず指定することになっており、内容を送信しない場合はFORM要素を使用せずに、ユーザーが入力や選択などの操作をするための要素(INPUT要素・BUTTON要素・SELECT要素・TEXTAREA要素)を直接配置することになっています。ただし、現状ではそのようにするとNetscape Navigatorでは入力フィールドやボタン、メニューなどが表示されなくなります。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>入力フォーム・サンプル</TITLE>
</HEAD>
<BODY>

<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
<LABEL>お名前:<INPUT type="text" name="namae"></LABEL>
<BR>
<LABEL>メール:<INPUT type="text" name="email"></LABEL>
</P>

<P>
<INPUT type="radio" name="sex" value="Male">男性
<INPUT type="radio" name="sex" value="Female">女性
</P>

<P>
<INPUT type="submit" value="送信">
<INPUT type="reset" value="クリア">
</P>
</FORM>

</BODY>
</HTML>
```

「入力フォーム」の「送信ボタン・内容のクリアボタンを作る」:P.148参照
「入力フォーム」の「ボタンを作る」:P.152参照

HTML

メールで送信するフォームを作る

**<FORM action="mailto:*メールアドレス*" method="post"
enctype="*MIMEタイプ*">～</FORM>**

URL	メールの送信先(mailto:～)
MIMEタイプ	内容を送信する際のMIMEタイプ

Internet Explorer

お名前: 山田 花子
メール: hana@yamapon.com
ご感想:
シンプルでなかなかいいです
送信 クリア

Netscape Navigator

お名前: 山田 花子
メール: hana@yamapon.com
ご感想:
シンプルでなかなかいいです
送信 クリア

【受信例】

enctype="application/x-www-form-urlencoded"

```
Hana, 9:35 PM 1999.8.7 +0900,Form posted from Mozilla
Subject:  Form posted from Mozilla

Date: Sat, 07 Aug 1999 21:35:46 +0900
From: Hana <hana@yamapon.com>
Organization: YAMAPON, Inc.
X-Mailer: Mozilla 4.05 (Macintosh; I; PPC)
To: norica@elfish.com
Subject: Form posted from Mozilla
Content-type: application/x-www-form-urlencoded
X-UIDL: ff65632e2dd2627a5fd631548f2e05f7

namae=%8ER%93c%81%40%89%D4%8Eq&email=hana%40yamapon.com&
%83V%83%93%83v%83%8B%82%C5%82%C8%82%A9%82%C8%82%A9%82%A2
```

enctype="text/plain"

```
Hana, 9:36 PM 1999.8.7 +0900,Form posted from Mozilla
Subject:  Form posted from Mozilla

Date: Sat, 07 Aug 1999 21:36:02 +0900
From: Hana <hana@yamapon.com>
Organization: YAMAPON, Inc.
X-Mailer: Mozilla 4.05 (Macintosh; I; PPC)
To: norica@elfish.com
Subject: Form posted from Mozilla
Content-type: text/plain
Content-Disposition: inline; form-data
X-UIDL: b812fa582a299cf0d223be75214cb89d

namae=山田　花子
email=hana@yamapon.com
kansou=シンプルでなかなかいいです。
```

enctype="multipart/form-data"

```
Hana, 9:35 PM 1999.8.7 +0900,Form posted from Mozilla
Subject:  Form posted from Mozilla

Date: Sat, 07 Aug 1999 21:35:54 +0900
From: Hana <hana@yamapon.com>
Organization: YAMAPON, Inc.
X-Mailer: Mozilla 4.05 (Macintosh; I; PPC)
To: norica@elfish.com
Subject: Form posted from Mozilla
Content-type: multipart/form-data; boundary=---------------------------
10760851715045
X-UIDL: a7e641fcb3179434029ff63b0a15737a

-----------------------------10760851715045
Content-Disposition: form-data; name="namae"

山田　花子
-----------------------------10760851715045
Content-Disposition: form-data; name="email"

hana@yamapon.com
-----------------------------10760851715045
Content-Disposition: form-data; name="kansou"

シンプルでなかなかいいです。
-----------------------------10760851715045--
```

解説

入力フォームの内容は、メールとして送信することもできます。
その場合は、action属性に「mailto:メールアドレス」を指定します。enctype属性を指定しない場合には、デフォルトの値としてMIMEタイプに「application/x-www-form-urlencoded」が採用されます。この場合は、送られてくるメールはそのままでは内容を読むことができない状態になっていますので、それを変換するためのソフトウェアが必要になります。enctype属性に「text/plain」または「multipart/form-data」を指定すると、メーラーでそのまま読める状態で送信することができます。
ただし、フォームのデータを受信するメーラーの種類やその設定、送信するブラウザがフォームの内容をメールとして送信する機能を備えているかどうかによっては、うまく機能しない場合もあります。確実にデータを送信して欲しいのであれば、メールで送信する形式にはせずに、CGIを利用した方がよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フォームをメールで送信するサンプル</TITLE>
</HEAD>
<BODY>

<FORM action="mailto:norica@elfish.com" method="post"
 enctype="text/plain">
<P>
<LABEL>お名前:<INPUT type="text" name="namae"></LABEL>
<BR>
<LABEL>メール:<INPUT type="text" name="email"></LABEL>
</P>

<P>
ご感想:<BR>
<TEXTAREA name="kansou" rows="3" cols="40">ここにご感想をどうぞ。
</TEXTAREA>
</P>

<P>
<INPUT type="submit" value="送信">
<INPUT type="reset" value="クリア">
</P>
</FORM>

</BODY>
</HTML>
```

HTML

1行のテキスト入力フィールドを作る

<INPUT type="text" name="*名前*" value="*デフォルト文字*" size="*幅*" maxlength="*最大文字数*">

名前	入力フィールドの名前
デフォルト文字	あらかじめ入力されている文字
幅	入力フィールドの幅(文字数)
最大文字数	入力可能な最大の文字数

Internet Explorer

お名前: 秀和 太郎

Netscape Navigator

お名前: 秀和　華子

解説

1行のテキスト入力フィールドを作成します。
name属性で指定する名前は、フォームの内容を受信した時にデータを見分けるためなどに使用されます。
この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>1行のテキスト入力フィールド・サンプル</TITLE>
</HEAD>
<BODY>
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
お名前:<INPUT type="text" name="namae" size="30">
</P>
</FORM>
</BODY>
</HTML>
```

「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照
「アクセシビリティ」の「ショートカットキーを割り当てる」:P.241参照

HTML

複数行のテキスト入力フィールドを作る

<TEXTAREA name="*名前*" rows="*行数*" cols="*幅*">〜</TEXTAREA>
<TEXTAREA wrap="*改行方法*" ・・・>〜</TEXTAREA>

名前	入力フィールドの名前	
行数	入力フィールドの行数	
幅	入力フィールドの幅(文字数)	
改行方法	off	改行しない
	soft	改行して表示
	hard	改行して送信

Internet Explorer

ご感想:
ここにご感想をどうぞ。

Netscape Navigator

ご感想:
ここにご感想をどうぞ。

解説

複数行のテキスト入力フィールドを作成します。
<TEXTAREA>〜</TEXTAREA>の範囲に記述した文字は、このフィールドにあらかじめ入力された状態で表示されます。name属性で指定する名前は、フォームの内容を受信した時にデータを見分けるためなどに使用されます。rows属性とcols属性は、入力フィールドの大きさを決定するもので、必ず指定することになっています。
wrap属性で指定する改行方法にsoftまたはhardを指定すると、入力した文字がフィールドの右端まで達した段階で自動的に改行されます。softの場合は画面上でのみ改行を行い、送信されるデータは改行されませんが、hardの場合には実際に送信されるデータも改行された状態になります。ただし、このwrap属性はHTML4.0で正式に定義されているものではなく、Netscape NavigatorとInternet Explorerが独自に拡張したものですので、注意してください。
この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>複数行のテキスト入力フィールド・サンプル</TITLE>
</HEAD>
<BODY>
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
ご感想：<BR>
<TEXTAREA name="kansou" rows="4" cols="40">ここにご感想をどうぞ。
</TEXTAREA>
</P>
</FORM>
</BODY>
</HTML>
```

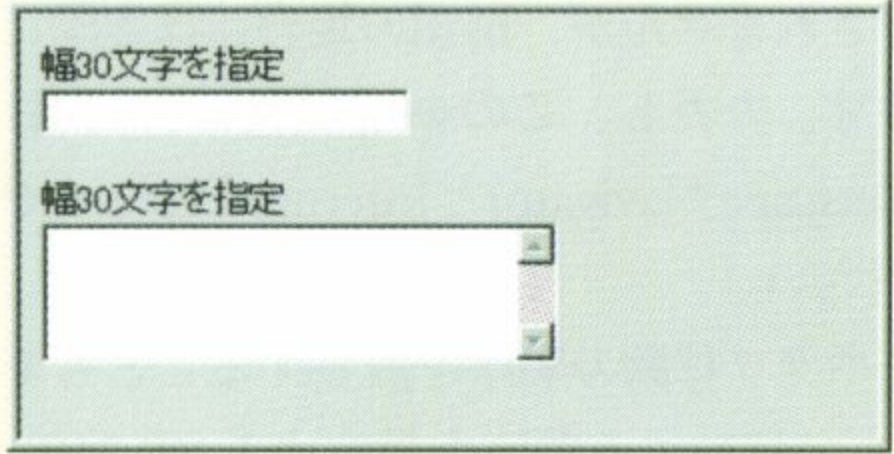
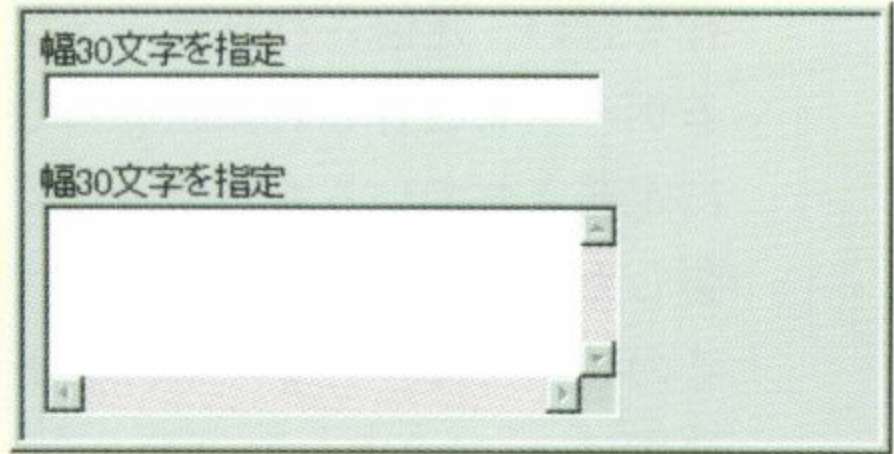
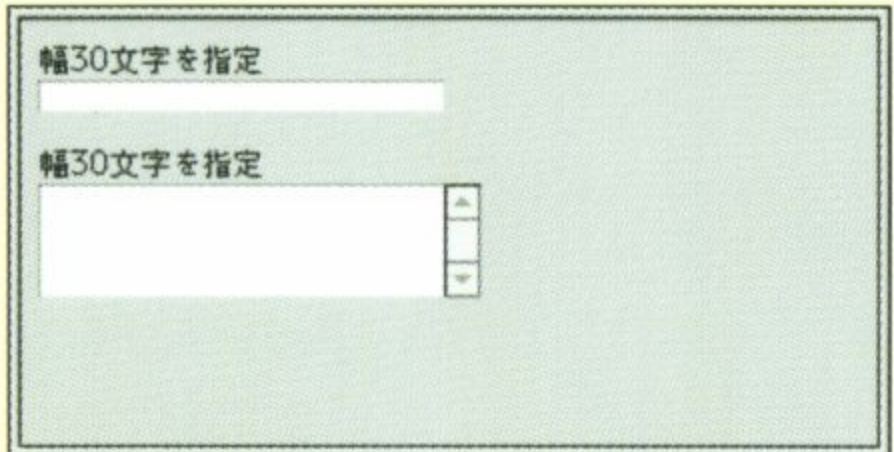
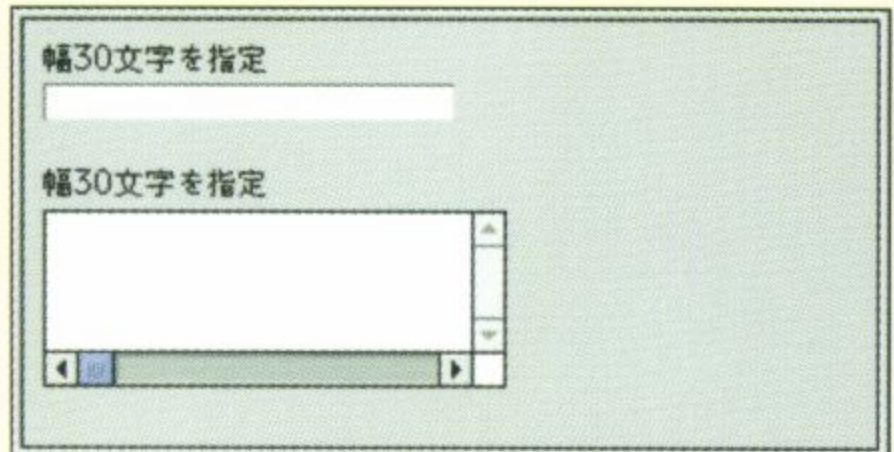
「アクセシビリティ」の「タブで移動する順序を指定する」：P.240参照
「アクセシビリティ」の「ショートカットキーを割り当てる」：P.241参照

注意

テキスト入力フィールドの大きさの違い

　フォームを構成する入力・選択項目は、環境によってその大きさが異なります。特にテキスト入力フィールドについては、その幅に大きな違いが見られます。

　フォームを作成する場合には、この点を考慮した上で、特定の環境に依存したものとならないように注意してください。

Windows版 Internet Explorer

Windows版 Netscape Navigator

Macintosh版 Internet Explorer

Macintosh版 Netscape Navigator

パスワードの入力フィールドを作る

<INPUT type="password" name="*名前*" value="*デフォルト文字*" size="*幅*" maxlength="*最大文字数*">

名前	入力フィールドの名前
デフォルト文字	あらかじめ入力されている文字
幅	入力フィールドの幅(文字数)
最大文字数	入力可能な最大の文字数

Internet Explorer

パスワード: *****

Netscape Navigator

パスワード: *****

解説

パスワードの入力に使用する1行のテキスト入力フィールドを作成します。
このフィールドに入力した文字は、他の文字や記号などに置き換えられて表示されます。name属性で指定する名前は、フォームの内容を受信した時にデータを見分けるためなどに使用されます。
この要素には、ショートカットキーやタブ移動の順序を設定することもできます。
詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>パスワードの入力フィールド・サンプル</TITLE>
</HEAD>
<BODY>
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
パスワード:<INPUT type="password" name="pw" size="20">
</P>
</FORM>
</BODY>
</HTML>
```

「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照
「アクセシビリティ」の「ショートカットキーを割り当てる」:P.241参照

HTML

隠しフィールドを作る

```
<INPUT type="hidden" name="名前" value="送信文字">
```

名前	隠しフィールドの名前
送信文字	送信する文字

Internet Explorer

お名前: shuwa jiro
メール: jiro@shuwa.ne.jp
男性　女性
送信　クリア

Netscape Navigator

お名前: shuwa akiko
メール: aki@shuwa.ne.jp
男性　女性
送信　クリア

解説

画面上には表示されないフィールドを作成します。
一般に、ユーザーに見せる必要のない特定の値をCGIプログラムに送信したい場合などに使用します。value属性で指定した内容が、固定値として送信されます。name属性で指定する名前は、フォームの内容を受信した側がこのデータを見分けるために使用されます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>隠しフィールド・サンプル</TITLE>
</HEAD>
<BODY>

<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
<INPUT type="hidden" name="recipient" value="norica@elfi
sh.com">
<INPUT type="hidden" name="subject" value="ユーザー登録">
<LABEL>お名前:<INPUT type="text" name="namae"></LABEL><BR>
<LABEL>メール:<INPUT type="text" name="email"></LABEL>
</P>
```

```
<P>
<INPUT type="radio" name="sex" value="Male">男性
<INPUT type="radio" name="sex" value="Female">女性
</P>

<P>
<INPUT type="submit" value="送信">
<INPUT type="reset" value="クリア">
</P>
</FORM>

</BODY>
</HTML>
```

NN2.0
NN3.0
NN4.0

IE3.0
IE4.0
IE5.0

HTML

送信ボタン・内容のクリアボタンを作る

```
<INPUT type="submit" value="ラベル" name="名前">  ←送信ボタン
<INPUT type="reset" value="ラベル">              ←クリアボタン
```

ラベル	ボタン上に表示される文字
名前	ボタンの名前

Internet Explorer

デフォルトの状態:
クエリ送信 リセット

ラベルを指定した状態:
送信 クリア

Netscape Navigator

デフォルトの状態:
Submit Query Reset

ラベルを指定した状態:
送信 クリア

解説

送信ボタンとクリアボタンは、多くの場合2つセットで使用されます。送信ボタンは、FORM要素の設定にしたがってフォーム内のデータを送信します。クリアボタンは、フォームのすべての内容を初期値に戻すために使用されます。
value属性でラベルを指定すると、ボタン上に表示される文字をそれぞれ設定することができます。ラベルを特に指定しなかった場合には、ブラウザがデフォルトの文字を表示させます(デフォルトの文字はブラウザによって異なります)。name属性で示す名前は、別の処理をさせたい複数の送信ボタンを配置する場合に、受信側でどの送信ボタンが押されたかを見分けるために利用されます。
この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>送信ボタンと内容のクリアボタン・サンプル</TITLE>
</HEAD>
<BODY>

<FORM action="/cgi-bin/formmail.cgi" method="post">
```

フォームを作る

```
<P>
デフォルトの状態：<BR>
<INPUT type="submit">
<INPUT type="reset">
</P>
<HR>
<P>
ラベルを指定した状態：<BR>
<INPUT type="submit" value="送信">
<INPUT type="reset"  value="クリア">
</P>

</FORM>

</BODY>
</HTML>
```

「アクセシビリティ」の「タブで移動する順序を指定する」：P.240参照
「アクセシビリティ」の「ショートカットキーを割り当てる」：P.241参照

HTML

画像による送信ボタンを作る

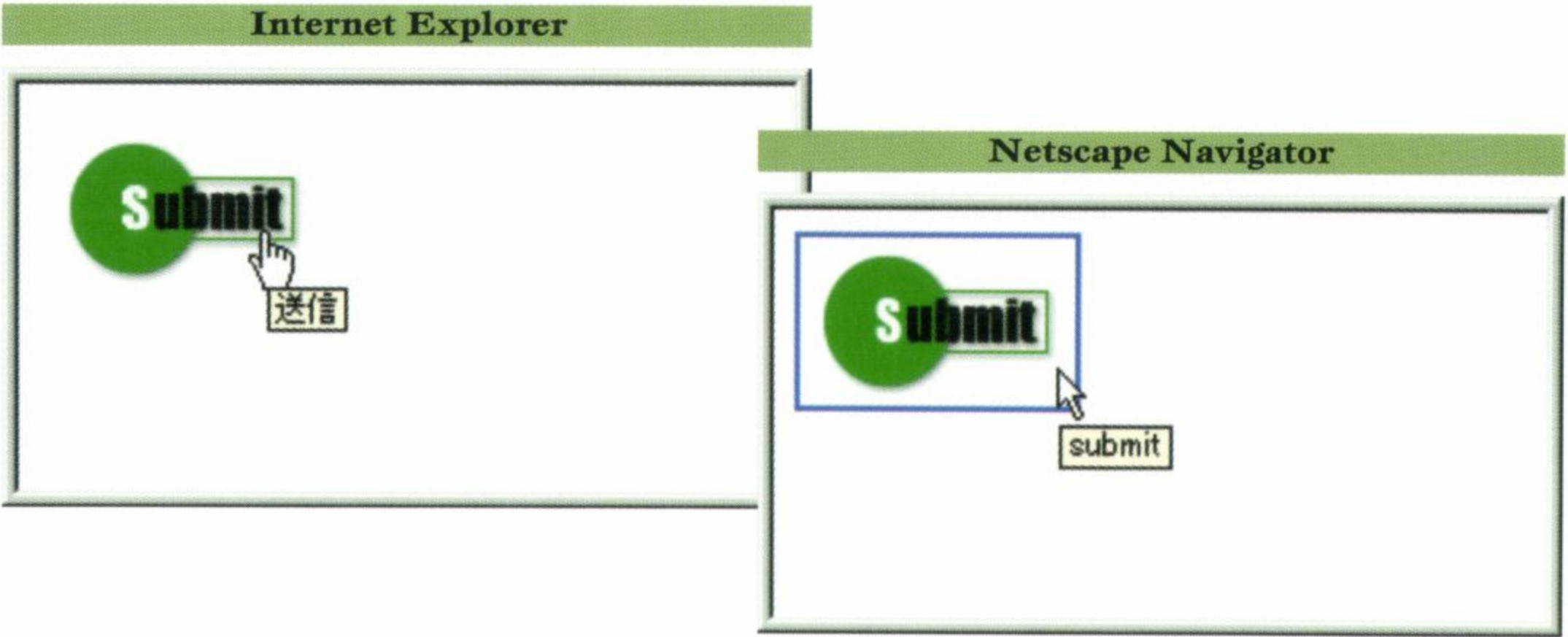
```
<INPUT type="image" src="URL" name="名前" alt="代替文字"
 align="位置">
```

URL	ボタンとして使用する画像ファイルのURL	
名前	ボタンの名前	
代替文字	画像のかわりとなるテキスト	
位置	top	画像の上とテキストの上を揃える
	middle	画像の中心とテキストのベースラインを揃える
	bottom	画像の下とテキストのベースラインを揃える(デフォルト)
	left	画像を左に配置し、右側に文字を回り込ませる
	right	画像を右に配置し、左側に文字を回り込ませる

Internet Explorer

Submit
送信

Netscape Navigator

Submit
submit

解説

通常、送信ボタンには<INPUT type="submit">を使用しますが、画像を送信ボタンとして機能させることができます。画像による送信ボタンを使用した場合には、フォームの内容と共に、画像のクリックされた位置(座標)も送信されます。

alt属性はHTML4.0で新しく採用された属性で、画像が表示できない場合に表示させるテキストを指定します。ただし、多くのブラウザはまだこの属性をサポートしておらず、name属性やvalue属性の値を代替文字として利用しているようです。

align属性は、通常の画像の場合と同様に表示位置を設定します。テキストが回り込むように設定した場合は、<BR clear="解除方向">で解除できます。ただし、この属性は廃止予定となっていますので、スタイルシートを使用した方がよいでしょう。

この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>画像による送信ボタン・サンプル</TITLE>
</HEAD>
<BODY bgcolor="white">
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
<INPUT type="image" src="submit.gif" name="submit"
 alt="送信">
</P>
</FORM>
</BODY>
</HTML>
```

「画像とマルチメディア」の「画像への回り込みを解除する」:P.130参照

「スタイルシート」の「縦方向の位置関係を指定する」:P.208参照

「スタイルシート」の「左右への配置と回り込みを指定する」:P.230参照

「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照

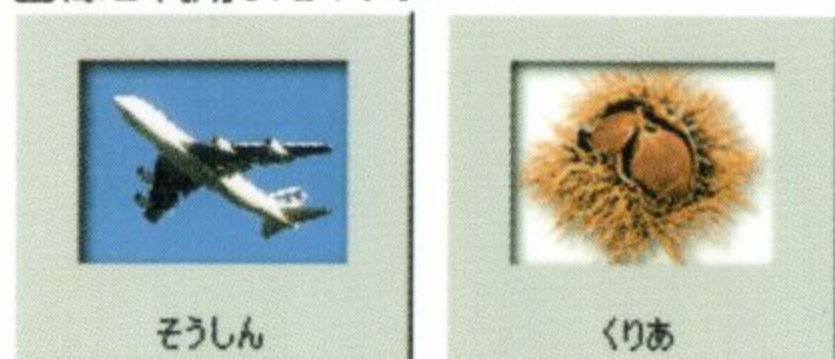
「アクセシビリティ」の「ショートカット・キーを割り当てる」:P.241参照

ボタンを作る

<BUTTON type="*タイプ*" name="*名前*" value="*送信文字*">～</BUTTON>

IE4.0 IE5.0

タイプ	submit	送信ボタン(デフォルト)
	reset	リセットボタン
	button	汎用ボタン
名前	ボタンの名前	
送信文字	送信する文字	

Internet Explorer

Netscape Navigator

解説

この要素は、HTML4.0から採用されたボタン専用の要素です。
type属性で指定する値によって、送信ボタン・リセットボタン・汎用ボタンのそれぞれの機能を果たすことができます。このボタンの特徴は、<BUTTON>～</BUTTON>の範囲の内容をボタンのラベルとして表示できることです。つまり、強調された文字や画像などをボタン上に表示させることができるわけです。name属性とvalue属性で示す値は、別の処理をさせたい複数の送信ボタンを配置する場合に、受信側でどの送信ボタンが押されたかを見分けるためなどに利用されます。
この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>ボタン・サンプル</TITLE>
</HEAD>
<BODY bgcolor="white">

<FORM action="/cgi-bin/formmail.cgi" method="post">

<P>
テキストを利用したボタン：<BR>
<BUTTON type="submit">
<FONT size="6"><B>送信</B></FONT>
</BUTTON>
<BUTTON type="reset">
<FONT size="6">クリア</FONT>
</BUTTON>
</P>
<HR>
<P>
画像を利用したボタン：<BR>
<BUTTON type="submit">
<BR><IMG src="hikouki.gif" width="92" height="69" alt="">
<BR>そうしん
</BUTTON>
<BUTTON type="reset">
<BR><IMG src="kuri.gif" width="92" height="69" alt="">
<BR>くりあ
</BUTTON>
</P>

</FORM>

</BODY>
</HTML>
```

「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照
「アクセシビリティ」の「ショートカットキーを割り当てる」:P.241参照

NN2.0
NN3.0
NN4.0
IE3.0
IE4.0
IE5.0

スクリプトで利用するボタンを作る

HTML

<INPUT type="button" name="*名前*" value="*ラベル*">

名前	ボタンの名前
ラベル	ボタン上に表示される文字

Internet Explorer

下のボタンをクリックすると、前のページに戻ります。
戻る

Netscape Navigator

下のボタンをクリックすると、前のページに戻ります。
戻る

解説

送信もリセットもしない汎用のボタンを作成します。
一般的には、onClickなどの属性を使用して、JavaScriptなどのスクリプト言語とあわせて利用されます。
この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>スクリプトで利用するボタン・サンプル</TITLE>
</HEAD>
<BODY>
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
下のボタンをクリックすると、前のページに戻ります。<BR>
<INPUT type="button" value="戻る" onClick="history.go(-1)">
</P>
</FORM>
</BODY>
</HTML>
```

「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照
「アクセシビリティ」の「ショートカットキーを割り当てる」:P.241参照

HTML

ラジオボタンを作る

```
<INPUT type="radio" name="名前" value="送信文字">
<INPUT type="radio" name="名前" value="送信文字" checked>
```

名前	ラジオボタンの名前
送信文字	選択された結果として送信される文字
checked	あらかじめ選択された状態にする場合に指定

Internet Explorer

■この本の内容はどうですか？
最高 良い 普通 悪い 最悪

Netscape Navigator

■この本の内容はどうですか？
最高 良い 普通 悪い 最悪

解説

ラジオボタンを作成します。
ラジオボタンは、複数の選択項目のうち1つだけ選択できる形式のボタンです。共通の項目に対する選択肢として使用するラジオボタンは、すべて同じ名前を指定する必要があります。また、データが送信された時にどの項目が選択されたのかを判別するために、value属性は必ず指定するようにしてください。
この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
■この本の内容はどうですか？<BR>
<INPUT type="radio" name="book" value="best">最高
<INPUT type="radio" name="book" value="good">良い
<INPUT type="radio" name="book" value="normal" checked>普通
<INPUT type="radio" name="book" value="bad">悪い
<INPUT type="radio" name="book" value="worst">最悪
</P>
</FORM>
```

「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照
「アクセシビリティ」の「ショートカットキーを割り当てる」:P.241参照

チェックボックスを作る

HTML

```
<INPUT type="checkbox" name="名前" value="送信文字">
<INPUT type="checkbox" name="名前" value="送信文字" checked>
```

名前	チェックボックスの名前
送信文字	選択された結果として送信される文字
checked	あらかじめ選択された状態にする場合に指定

Internet Explorer

■好きな色は？
☑白 ☑黒 □グレー □赤 □青 □黄

Netscape Navigator

■好きな色は？
☑白 □黒 □グレー □赤 ☑青 □黄

解説

チェックボックスを作成します。
チェックボックスは、複数の選択項目の中から該当する項目を複数選択できるようにする場合に使用します。共通の項目に対する選択肢として使用するチェックボックスは、すべて同じ名前を指定する必要があります。また、データが送信された時にどの項目が選択されたのかを判別するために、value属性は必ず指定するようにしてください。
この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
■好きな色は？<BR>
<INPUT type="checkbox" name="color" value="white"
 checked>白
<INPUT type="checkbox" name="color" value="black">黒
<INPUT type="checkbox" name="color" value="gray">グレー
<INPUT type="checkbox" name="color" value="red">赤
<INPUT type="checkbox" name="color" value="blue">青
<INPUT type="checkbox" name="color" value="yellow">黄
</P>
</FORM>
```

「アクセシビリティ」の「タブで移動する順序を指定する」：P.240参照
「アクセシビリティ」の「ショートカットキーを割り当てる」：P.241参照

HTML

メニューを作る

```
<SELECT name="名前">～</SELECT>              ←メニュー全体
<OPTION value="送信文字">～</OPTION>          ←メニュー項目
<OPTION selected>～</OPTION>                 ←メニュー項目
```

名前	メニューの名前
送信文字	選択された結果として送信される文字
selected	あらかじめ選択された状態にする場合に指定

Internet Explorer

あなたの年齢は次のどれに当てはまりますか？
30代

あなたの年齢は次のどれに当てはまりますか？
30代
10代
20代
30代
40代
50代

Netscape Navigator

あなたの年齢は次のどれに当てはまりますか？
30代

あなたの年齢は次のどれに当てはまりますか？
30代
10代
20代
30代
40代
50代

解説

メニューを作成します。
メニュー全体を<SELECT>～</SELECT>で囲んで示し、その中に選択肢を表す<OPTION>～</OPTION>を必要な数だけ配置します。<OPTION>～</OPTION>の範囲には、実際にメニューに表示される選択肢となるテキストを入れます。また、value属性を省略した場合には、ここに入れたテキスト自体が選択された項目として送信されます。
この要素には、タブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>メニュー・サンプル</TITLE>
</HEAD>
<BODY>
```

```
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
あなたの年齢は次のどれに当てはまりますか？<BR>
<SELECT name="年齢">
<OPTION>10代</OPTION>
<OPTION>20代</OPTION>
<OPTION selected>30代</OPTION>
<OPTION>40代</OPTION>
<OPTION>50代</OPTION>
</SELECT>
</P>
</FORM>
</BODY>
</HTML>
```

「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照

メニューの選択肢をグループ化する

```
<OPTGROUP label="グループ名">～</OPTGROUP> ←グループを作成
<OPTION label="短い選択肢">～</OPTION>     ←グループ内の項目
```

短い選択肢　　グループ名に対応させた短い選択肢

解説

SELECT要素で作成されるメニューの選択肢をグループ化します(結果としてメニューは階層メニューになります)。

OPTGROUP要素のlabel属性の値は、グループ(階層)名としてメニューに表示されるものですので、必ず指定するようにしてください。OPTION要素のlabel属性の値には、グループ名が表示されることによって省略できる部分を省いた、短い選択肢となるテキストを指定します。この属性を省略した場合には、<OPTION>～</OPTION>の範囲に指定されている内容がそのまま利用されます。

このメニューのグループ化はHTML4.0で新しく定義されたものですが、現状ではほとんどサポートされていません。しかし、サポートされていないブラウザでもOPTGROUP要素とlabel属性が無視されるだけで、今まで通りのメニューを表示させることができます。

Sample

```
<FORM action="cgi-bin/formmail.cgi" method="post">
<P>
あなたが最も多く利用しているブラウザはどれですか？<BR>
<SELECT name="使用ブラウザ">
<OPTGROUP label="Internet Explorer">
  <OPTION label="5.x">Internet Explorer 5.x</OPTION>
  <OPTION label="4.x">Internet Explorer 4.x</OPTION>
  <OPTION label="3.x">Internet Explorer 3.x</OPTION>
  <OPTION label="2.x">Internet Explorer 2.x</OPTION>
</OPTGROUP>
<OPTGROUP label="Netscape Navigator">
  <OPTION label="4.x">Netscape Navigator 4.x</OPTION>
  <OPTION label="3.x">Netscape Navigator 3.x</OPTION>
  <OPTION label="2.x">Netscape Navigator 2.x</OPTION>
</OPTGROUP>
<OPTGROUP label="Lynx">
  <OPTION label="3.x">Lynx 3.x</OPTION>
  <OPTION label="2.x">Lynx 2.x</OPTION>
  <OPTION label="1.x">Lynx 1.x</OPTION>
</OPTGROUP>
</SELECT>
</P>
</FORM>
```

リストボックスを作る

HTML

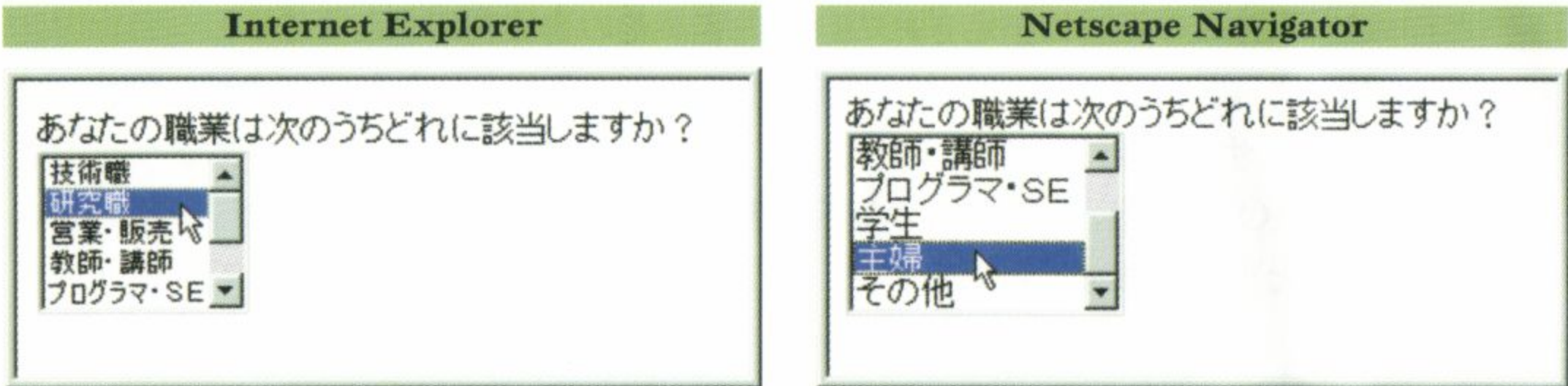
```
<SELECT size="行数" name="名前" multiple>～</SELECT>
                                        ←リストボックス
<OPTION value="送信文字">～</OPTION>    ←選択項目
<OPTION selected>～</OPTION>            ←選択項目
```

行数	リストボックスの表示行数
名前	リストボックスの名前
multiple	複数の項目を選択可能にする場合に指定
送信文字	選択された結果として送信される文字
selected	あらかじめ選択された状態にする場合に指定

Internet Explorer

あなたの職業は次のうちどれに該当しますか？
技術職
研究職
営業・販売
教師・講師
プログラマ・SE

Netscape Navigator

あなたの職業は次のうちどれに該当しますか？
教師・講師
プログラマ・SE
学生
主婦
その他

解説

メニューを作成するSELECT要素にsize属性を指定すると、リストボックスとして表示されます。
メニューの場合と同様に、リストボックス全体を<SELECT>～</SELECT>で囲んで示し、その中に選択肢を表す<OPTION>～</OPTION>を必要な数だけ配置します。<OPTION>～</OPTION>の範囲には、実際にリストボックスに表示される選択肢となるテキストを入れます。また、value属性を省略した場合には、ここに入れたテキスト自体が選択された項目として送信されます。OPTION属性で示される選択肢の数が、size属性で示される表示行数より多い場合には、リストボックスに自動的にスクロールバーが付けられます。
この要素には、タブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>リストボックス・サンプル</TITLE>
</HEAD>
```

```
<BODY>
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
あなたの職業は次のうちどれに該当しますか？<BR>
<SELECT size="5" name="職業" multiple>
<OPTION>技術職</OPTION>
<OPTION>研究職</OPTION>
<OPTION>営業・販売</OPTION>
<OPTION>教師・講師</OPTION>
<OPTION>プログラマ・ＳＥ</OPTION>
<OPTION>学生</OPTION>
<OPTION>主婦</OPTION>
<OPTION>その他</OPTION>
</SELECT>
</P>
</FORM>
</BODY>
</HTML>
```

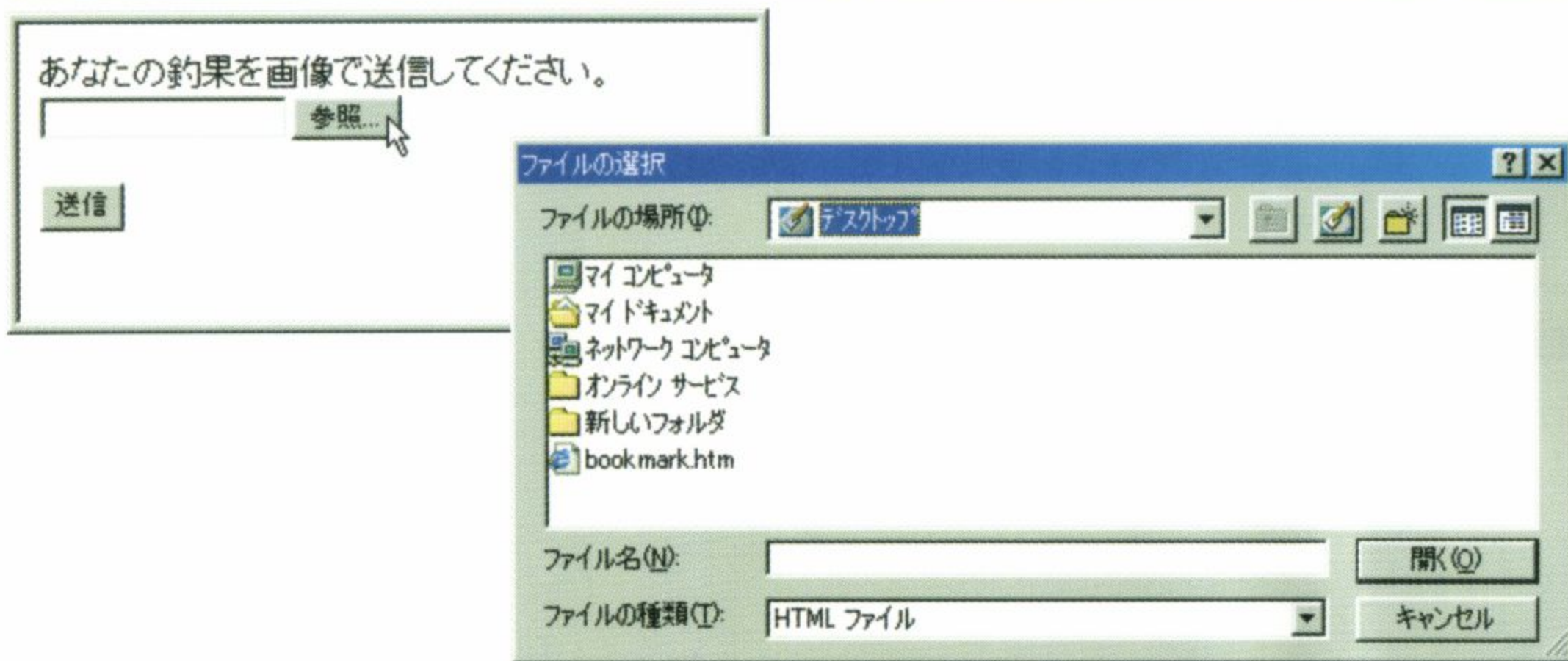
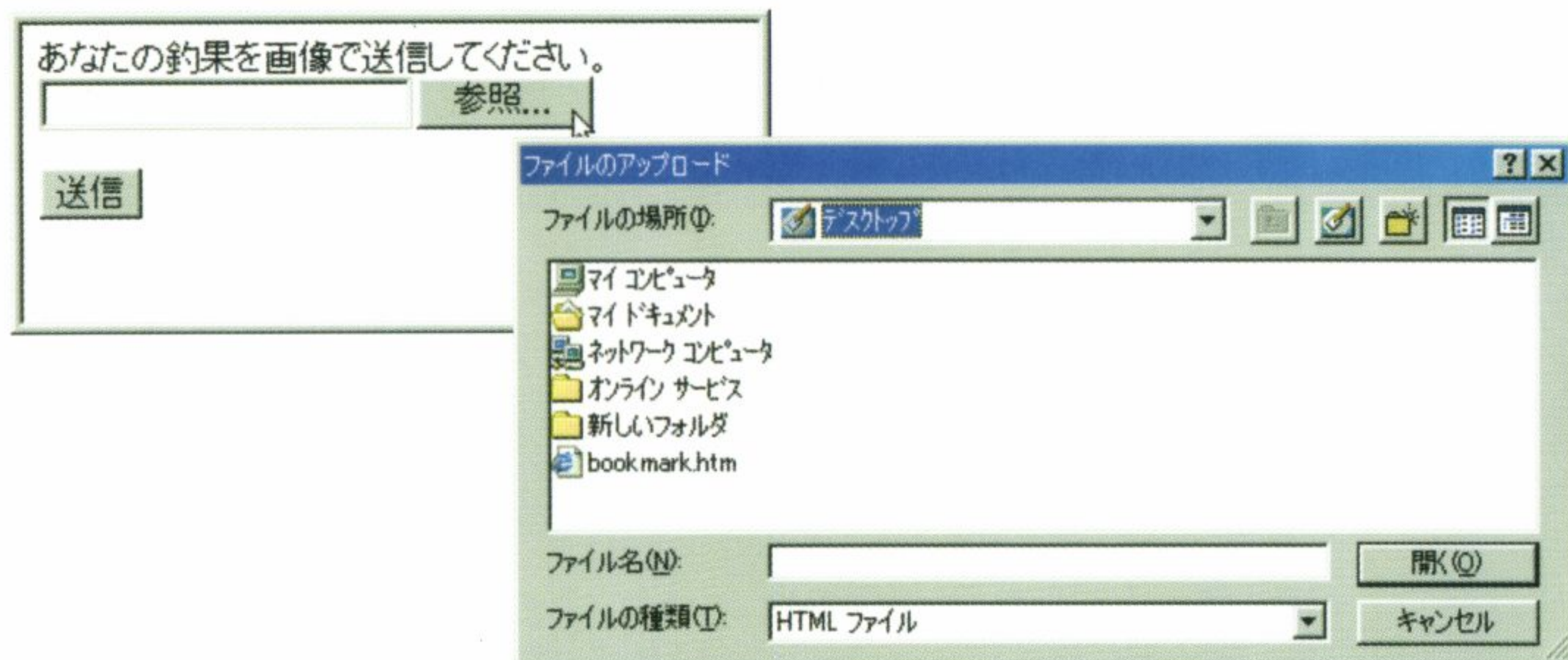
「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照

HTML

ファイルを選択する

```
<INPUT type="file" name="名前" value="ファイル名" accept="MIMEタイプ">
```

ファイル名	初期値として表示されるファイル名
MIMEタイプ	受け付け可能な「,」区切りのMIMEタイプリスト

Internet Explorer

Netscape Navigator

解説

フォームのデータとして送信するファイルを選択できるような、ボタンとフィールドを自動的に作成します。

accept属性には、受信プログラムが受け付けることのできるファイルの種類をMIMEタイプで指定します。複数の種類を受信可能な場合には、それらを「,」で区切って指定することができます。ただし、この属性は現状ではほとんどサポートされていないようです。

この要素には、ショートカットキーやタブ移動の順序を設定することもできます。詳細は、「アクセシビリティ」の「タブで移動する順序を指定する」(P.240)・「ショートカットキーを割り当てる」(P.241)を参照してください。

※この機能を利用する場合、FORM要素のenctype属性には「multipart/form-data」を、method属性には「post」を指定する必要があります。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>ファイルの送信・サンプル</TITLE>
</HEAD>
<BODY>

<FORM action="/cgi-bin/snap.cgi" enctype="multipart/
form-data" method="post">

<P>
あなたの釣果を画像で送信してください。<BR>
<INPUT type="file" name="imagefile" accept="image/jpeg,
image/gif">
</P>

<P>
<INPUT type="submit" value="送信">
</P>

</FORM>

</BODY>
</HTML>
```

「アクセシビリティ」の「タブで移動する順序を指定する」:P.240参照
「アクセシビリティ」の「ショートカットキーを割り当てる」:P.241参照

HTML

フォームの内容にラベルを付ける

<LABEL>～</LABEL>
<LABEL for="*参照ID*">～</LABEL>

IE4.0
IE5.0

参照ID	ラベルを付ける入力・選択項目のID

Internet Explorer

お名前: shuwa saburo
メール: saburo@shuwa.ne.jp
男性　女性

Netscape Navigator

お名前: shuwa reiko
メール: rei@shuwa.ne.jp
男性　女性

解説 フォーム内に含まれる入力項目や選択項目のうち、value属性によってラベルを付けることのできないものに対してラベルを設定します(つまり、入力・選択項目とテキストを関連付けます)。
これによって、例えばラジオボタンやチェックボックスは、文字部分に対するクリックにも反応するようになります。ラベルの付け方には、2通りあります。1つは、<LABEL>～</LABEL>の範囲内にラベルとなるテキストと入力・選択項目を含める方法です。2つめは、<LABEL>～</LABEL>の範囲内にはラベルとなるテキストのみを配置して、入力・選択項目で指定したIDと同じものをfor属性で指定する方法です。この場合は、ラベルと入力・選択項目が、必ず1対1になるようにしてください。
この要素には、ショートカットキーを設定することもできます。詳細は、「アクセシビリティ」の「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>ラベル・サンプル</TITLE>
</HEAD>
<BODY>

<FORM action="/cgi-bin/snap.cgi" enctype="multipart/
form-data" method="post" >
```

```
<P>
<LABEL for="nm">お名前:</LABEL>
<INPUT type="text" name="namae" id="nm"><BR>
<LABEL for="em">メール:</LABEL>
<INPUT type="text" name="email" id="em">
</P>

<P>
<LABEL><INPUT type="radio" name="sex" value="Male">男性
</LABEL>
<LABEL><INPUT type="radio" name="sex" value="Female">女性
</LABEL>
</P>

</FORM>

</BODY>
</HTML>
```

「アクセシビリティ」の「ショートカットキーを割り当てる」:P.241参照

コラム

ラベルについて

普通のアプリケーションを利用している場合、ラジオボタンやチェックボックスは、テキスト部分をクリックしても反応します。しかし、このLABEL要素ができるまでは、Webページ上のラジオボタンやチェックボックスは、そのようには動作しませんでした。つまり、ボタン部分とテキスト部分の関連を明確に示す方法がなかったため、それらが1つのものとして認識されず、ボタン部分をクリックしなければ何も反応しない状態になっていたのです。

LABEL要素をサポートしているブラウザはまだ多くはありませんが、上記のような不自然な動きをさせないためにも、必ず指定しておいた方がよいでしょう。

余談ですが、ラベルを付ける方法が2つある理由についても述べておきます。一般的には、入力・選択項目とテキスト全体を<LABEL>~</LABEL>で囲う方法が分かりやすくていいと思います。しかし、現実的にフォームをキレイに配置しようとした場合、入力・選択項目とテキストは、別々の段落になったり、テーブル内の別のセルに配置されたりすることも多いはずです。そのような場合には、for属性を利用して関連付けを行うことになります。

フォームの内容をグループ化する

<FIELDSET>～</FIELDSET>　←グループ化
<LEGEND align="*位置*">～</LEGEND>　←グループのタイトル

【位置】

top	タイトルを上に表示する(デフォルト)
bottom	タイトルを下に表示する
left	タイトルを左に表示する
right	タイトルを右に表示する

Internet Explorer

個人情報
名前:　住所:
電話:
会社情報
社名:　住所:
電話:

Netscape Navigator

個人情報
名前:　住所:
電話:
会社情報
社名:　住所:
電話:

解説

FIELDSET要素は、フォームに含まれる入力項目や選択項目をグループ化します。<FIELDSET>～</FIELDSET>の範囲の1番始めには、LEGEND要素を指定してそのグループのタイトルを付けます。
align属性は廃止予定となっていますが、現在のところ、これに代わるスタイルシートのプロパティは定義されていません。
LEGEND要素には、ショートカットキーを設定することもできます。詳細は、「アクセシビリティ」の「ショートカットキーを割り当てる」(P.241)を参照してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フォームのグループ化サンプル</TITLE>
</HEAD>
<BODY>

<FORM action="/cgi-bin/snap.cgi" enctype="multipart/
form-data" method="post" >

<FIELDSET>
<LEGEND>個人情報</LEGEND>
<P>
名前：<INPUT type="text" name="uname">
住所：<INPUT type="text" name="uaddrs" size="36"><BR>
電話：<INPUT type="text" name="uphone">
</P>
</FIELDSET>

<FIELDSET>
<LEGEND>会社情報</LEGEND>
<P>
社名：<INPUT type="text" name="cname">
住所：<INPUT type="text" name="caddrs" size="36"><BR>
電話：<INPUT type="text" name="cphone">
</P>
</FIELDSET>

</FORM>

</BODY>
</HTML>
```

「アクセシビリティ」の「ショートカットキーを割り当てる」：P.241参照

HTML

フレームの全体構成を指定する

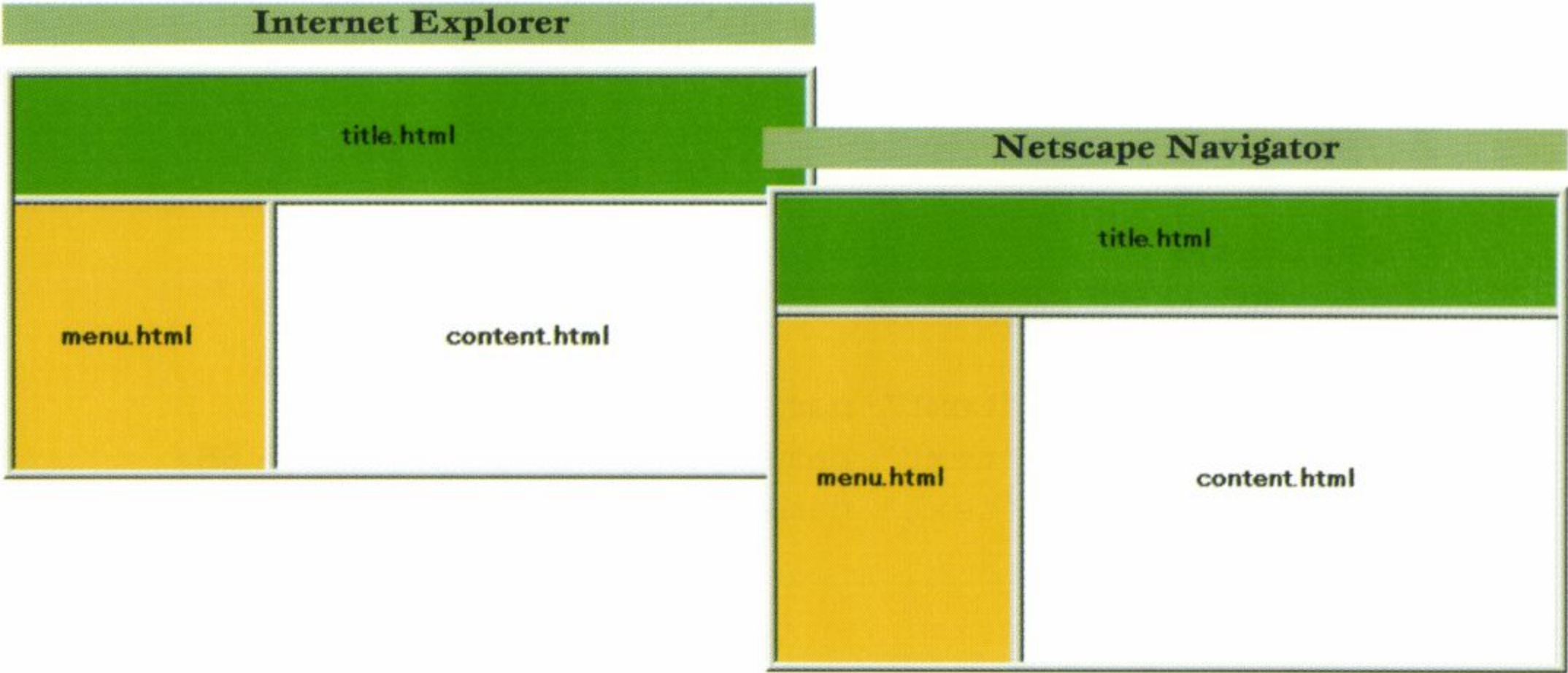
```
<FRAMESET rows="分割高さ">～</FRAMESET>
<FRAMESET cols="分割幅">～</FRAMESET>
<FRAME src="URL" name="フレーム名">
```

分割高さ	横割りした時の各高さを「,」区切りで指定(ピクセル・%・*)
分割幅	縦割りした時の各幅を「,」区切りで指定(ピクセル・%・*)
URL	フレームの内容として表示するHTML文書のURL
フレーム名	リンクなどの表示先として指定する場合に利用する名前

Internet Explorer

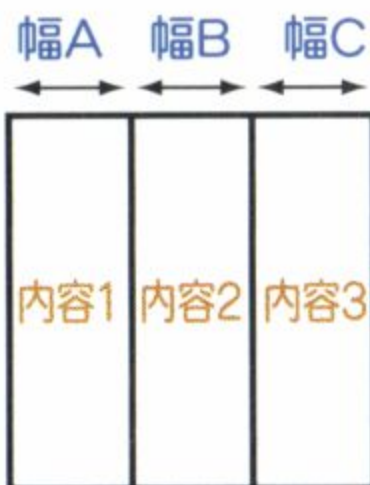
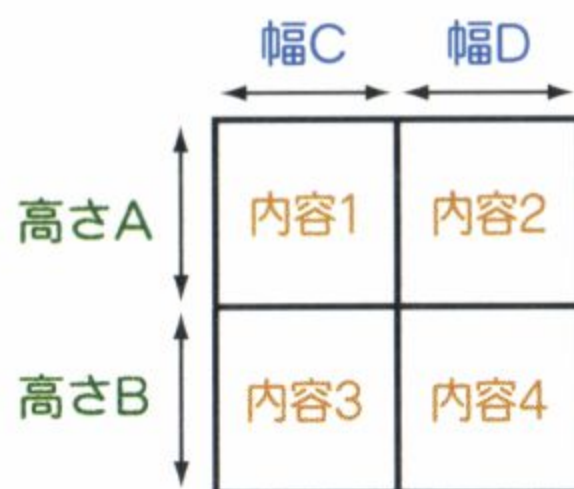
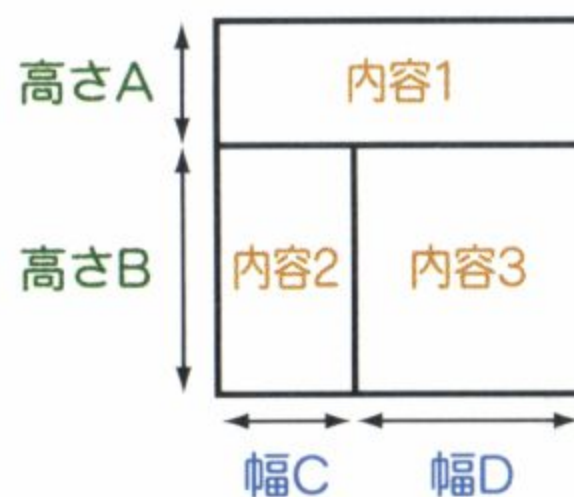
title.html
menu.html
content.html

Netscape Navigator

title.html
menu.html
content.html

解説

フレーム機能を利用すると、ウインドウを縦横に区切って、その中にそれぞれ別のHTML文書を表示させることができます。
ウインドウをどのように区切るかを指定するのがFRAMESET要素で、分割された各フレームに表示する内容(HTML文書)を指定するのがFRAME要素です。フレームを指定する文書では、本来BODY要素があるべき部分にFRAMESET要素を配置します。BODY要素は使用できませんので注意してください。ただし、後述する<NOFRAMES>～</NOFRAMES>(「フレームに未対応のブラウザ向けの内容を入れる」：P.176)内には必要です。

FRAMESET要素のrows属性は横割りするフレームの各高さを上から順に、cols属性は縦割りするフレームの各幅を左から順に「,」で区切って指定します。これによって、指定された数値の数だけ分割されます。
また、<FRAMESET>～</FRAMESET>の範囲には、作成されたフレームの個数分の内容を順に入れる必要があります。フレームをそれ以上分割しないのであればFRAME要素で読み込むHTML文書を指定し、更に分割する場合にはFRAMESET要素を配置して(入れ子にして)その部分の分割を指定します。

※フレームを定義するHTML文書では、<!DOCTYPE>に「HTML4.0 Frameset DTD」を指定することに注意してください。

```
<FRAMESET rows="高さA,高さB,高さC">
  <FRAME src=内容1.html">
  <FRAME src=内容2.html">
  <FRAME src=内容3.html">
</FRAMESET>
```

```
<FRAMESET cols="幅A,幅B,幅C">
  <FRAME src=内容1.html">
  <FRAME src=内容2.html">
  <FRAME src=内容3.html">
</FRAMESET>
```

```
<FRAMESET rows="高さA,高さB" cols="幅C,幅D">
  <FRAME src=内容1.html">
  <FRAME src=内容2.html">
  <FRAME src=内容3.html">
  <FRAME src=内容4.html">
</FRAMESET>
```

```
<FRAMESET rows="高さA,高さB">
  <FRAME src=内容1.html">
  <FRAMESET cols="幅C,幅D">
    <FRAME src=内容2.html">
    <FRAME src=内容3.html">
  </FRAMESET>
</FRAMESET>
```

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
 "http://www.w3.org/TR/REC-html40/frameset.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フレームの全体構成サンプル</TITLE>
</HEAD>

<FRAMESET rows="70,*">
  <FRAME src="title.html" name="logo">
  <FRAMESET cols="150,*">
    <FRAME src="menu.html" name="menu">
    <FRAME src="content.html" name="content">
  </FRAMESET>
```

```
    <NOFRAMES>
    <BODY>
      ~
    </BODY>
    </NOFRAMES>
</FRAMESET>

</HTML>
```

コラム「HTMLのバージョンと<!DOCTYPE>の書き方」:P.28参照
「フレーム」の「フレームに未対応のブラウザ向けの内容を入れる」:P.176参照

Tip 「*」による幅や高さの割合の指定

FRAMESET要素のrows・cols属性や、COL・COLGROUP要素のwidth属性では、その幅や高さを「ピクセル」や「パーセンテージ」で指定する他に、「*」を利用して指定することができます。

幅や高さが複数に分割されている場合、まず「ピクセル」や「パーセンテージ」で指定されているものがあれば、その幅(高さ)が確保されます。そして、残りの部分が「*」の前に付けられた数字の割合で分配されます(単に「*」を指定した場合は、「1*」と指定したものと解釈されます)。例えば、「*,2*,3*」と指定されていて分配可能な範囲が60ピクセルだった場合には、60ピクセルが6分割(1+2+3)されて、「*」は10ピクセル、「2*」は20ピクセル、「3*」は30ピクセルということになります。

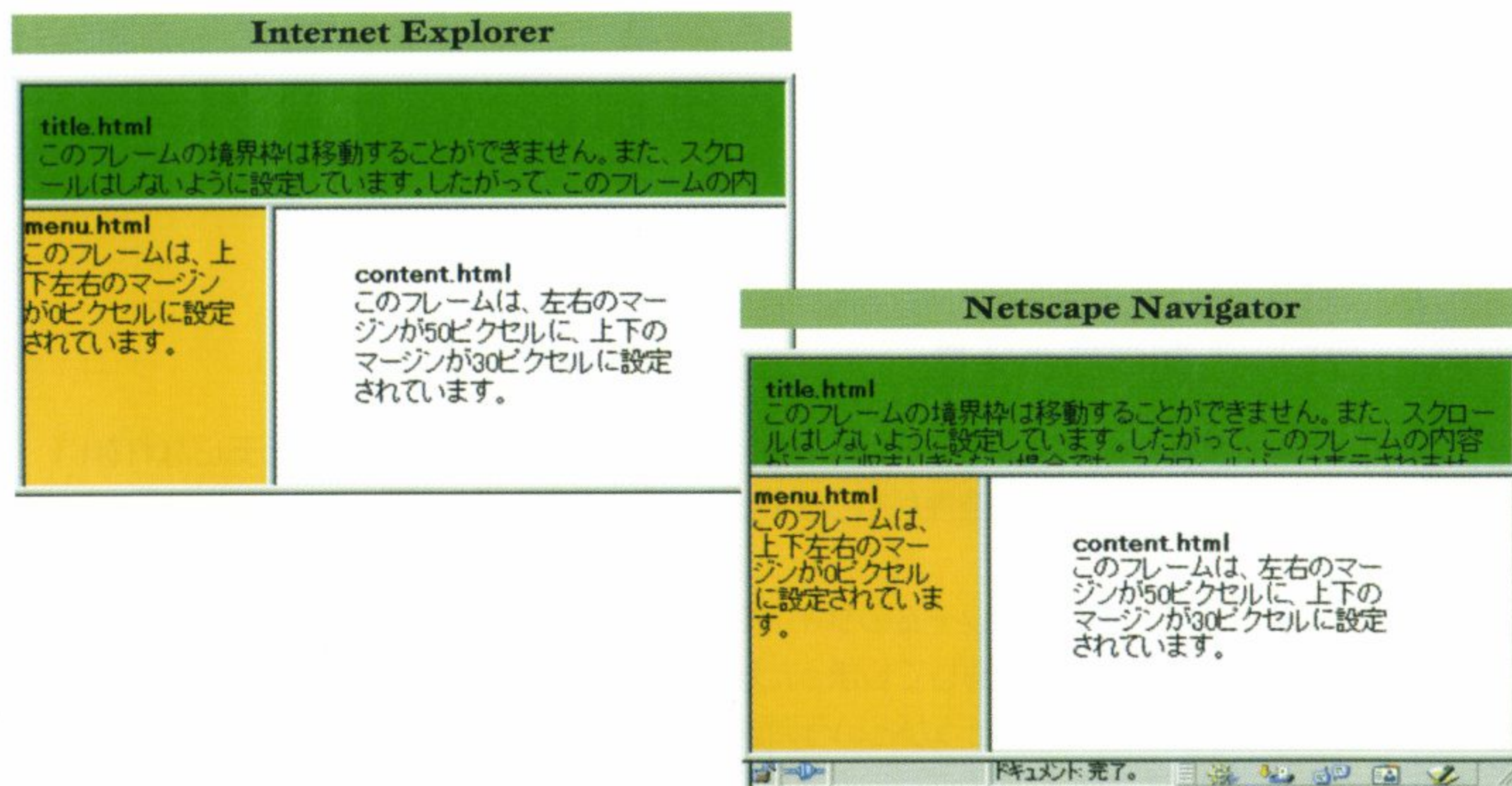
フレーム内での表示方法を設定する

HTML

<FRAME scrolling="スクロールの制御" noresize>
<FRAME marginwidth="左右のマージン" marginheight="上下のマージン">

スクロールの制御	auto	必要に応じてスクロール可能(デフォルト)
	yes	常にスクロール可能(スクロールバーを表示)
	no	常にスクロール不可(スクロールバーを非表示)
noresize	フレームのサイズ変更禁止	
左右のマージン	フレーム内の左右のマージン(ピクセル)	
上下のマージン	フレーム内の上下のマージン(ピクセル)	

Internet Explorer

Netscape Navigator

解説

一般的なブラウザでは、フレームを区切る境界枠はドラッグして移動できるようになっています。
noresize属性を指定すると、この境界枠の移動をできないようにして、フレームの表示領域を固定することができます。scrolling属性は、そのフレームの内容を表示する場合のスクロールに関する設定をします。marginwidth属性とmarginheight属性は、そのフレーム内のマージンを設定します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
 "http://www.w3.org/TR/REC-html40/frameset.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フレーム内での表示方法サンプル</TITLE>
</HEAD>
```

```
<FRAMESET rows="70,*">
  <FRAME src="title.html" scrolling="no" noresize>
  <FRAMESET cols="150,*">
    <FRAME src="menu.html" marginwidth="0" marginheight
="0">
    <FRAME src="content.html" marginwidth="50" marginheight
="30">
  </FRAMESET>
  <NOFRAMES>
  <BODY>
    ∫
  </BODY>
  </NOFRAMES>
</FRAMESET>

</HTML>
```

注意

フレーム利用時の注意

フレームは、様々な環境を知った上で上手に利用しないと、とても見る気になれないページになってしまう危険性があります。

まず、分割された各フレームのサイズを変更(フレーム枠を移動)できないようにしたり、フレームの枠自体を消してしまった場合、どの環境で見てもフレームの内容がそこにうまく収まっているかどうかという問題があります(スクロールも不可になっている場合は、更に深刻です)。

すべてのフレームの大きさをウインドウに対するパーセンテージで指定した場合、大きな画面では普通に見えても小さな画面では1部しか表示されず、どうにもならない場合があります。

また、文字のサイズは環境によってビックリするほど違っている場合がありますし、フォームで使われるメニューやテキストフィールドなども、環境によってその大きさがかなり違います。画像などの大きさが固定のものは別として、Web上の多くのものは大きさが常に同じではないことに注意してください(できれば様々な環境で確認してみることをお勧めします)。

その他にも、1度に複数の文書を読み込むためロードや再表示に時間がかかるということや、任意のページをブックマークすることができないなどの理由で、フレームが嫌いだというユーザーはかなりいます。以上のような点を考慮した上で、そのページにはフレームを使うべきかどうか、また、フレームの分割方法やサイズ、その他の指定は適切かどうかを確認してみてください。

フレームを区切る枠の表示／非表示を設定する

<FRAME frameborder="*枠の表示指定*">

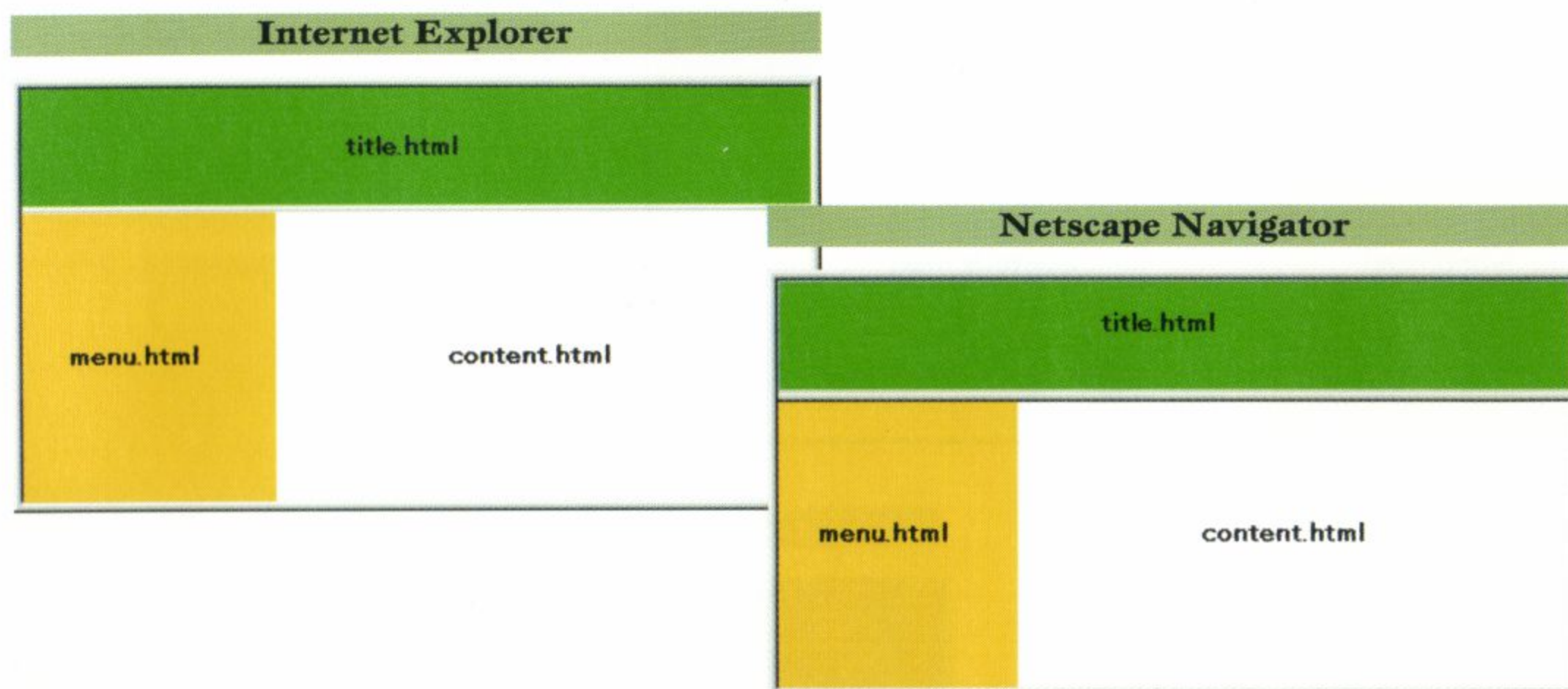
【枠の表示指定】

1	表示(デフォルト)
0	非表示

Internet Explorer

title.html
menu.html
content.html

Netscape Navigator

title.html
menu.html
content.html

解説

そのフレームと隣り合うフレームを区切る枠を表示するかどうかを設定します。一般的なブラウザでは、この属性をFRAMESET要素にも指定できますが、HTML 4.0で定義されているのはFRAME要素の属性だけです。ここでいう枠とは、一般的に立体的に表示される枠のことで、これを非表示に設定してもフレームとフレームの間の(平面的な)空間は残ります。
また、あるフレームの枠を非表示に設定していても、隣り合うフレームの枠が表示されるように設定されている場合には、その間の枠は表示されます。

Sample

```
<FRAMESET rows="70,*">
  <FRAME src="title.html">
  <FRAMESET cols="150,*">
    <FRAME src="menu.html" frameborder="0">
    <FRAME src="content.html" frameborder="0">
  </FRAMESET>
  <NOFRAMES>
  <BODY>
    ∫
  </BODY>
  </NOFRAMES>
</FRAMESET>
```

「フレーム」の「フレームを区切る枠を完全に消す」：次項参照

HTML

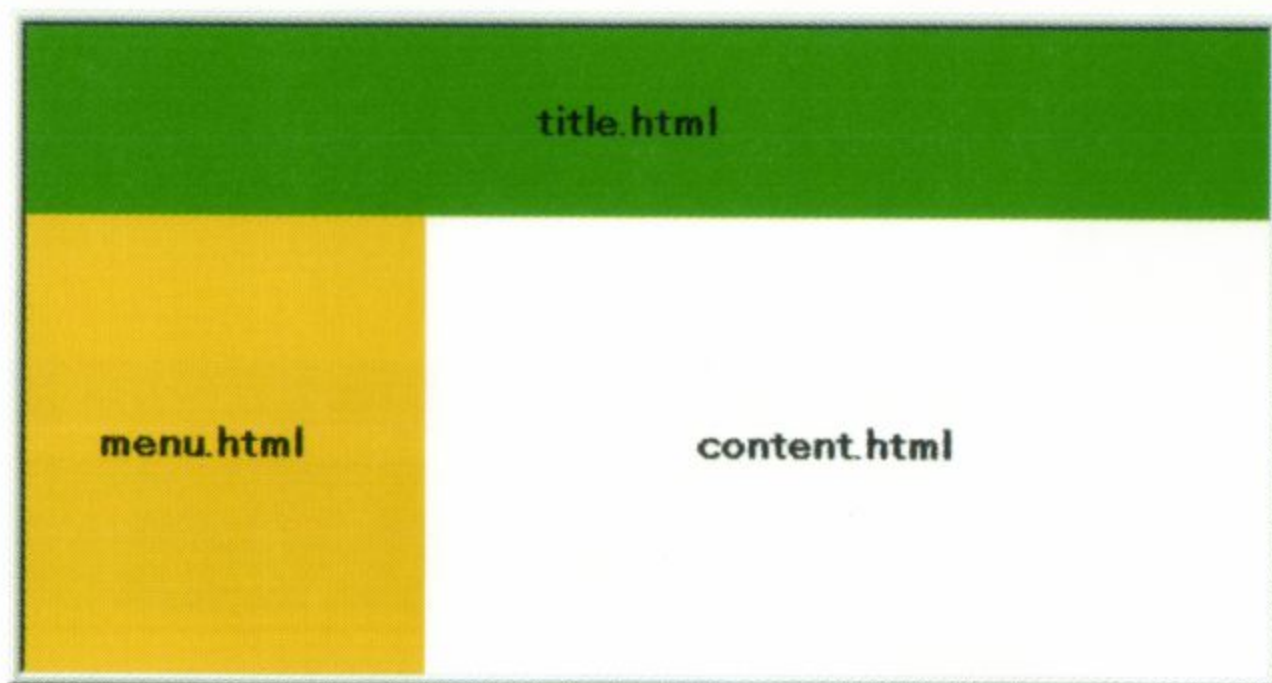
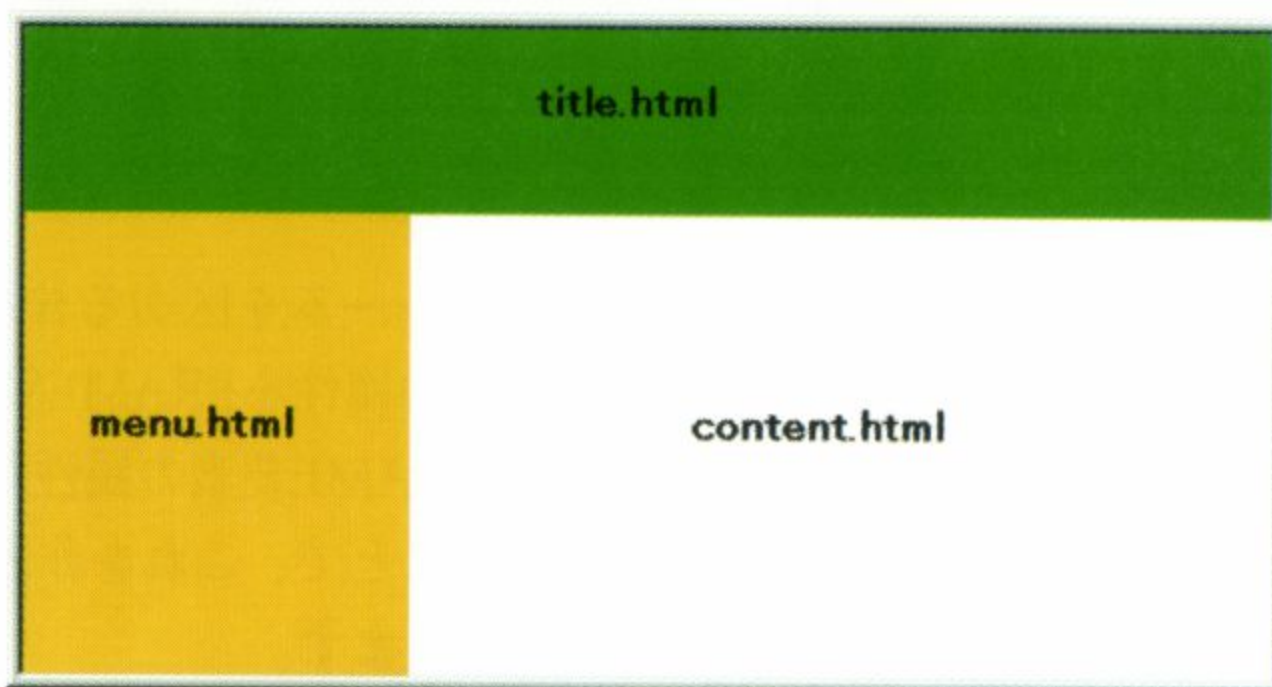
フレームを区切る枠を完全に消す

```
<FRAMESET frameborder="0" border="0" framespacing="0"> ~
</FRAMESET>
```

Internet Explorer

title.html
menu.html
content.html

Netscape Navigator

title.html
menu.html
content.html

解説

frameborder属性を利用するとフレームを区切る枠を非表示にすることはできますが、その場合はフレームとフレームの間の空間が残ってしまいます。この空間も含めて、枠を完全に消すためには、Netscape NavigatorとInternet Explorerで独自拡張された属性を利用する必要があります。
具体的には、frameborder属性に加えて、Netscape Navigatorのborder属性とInternet Explorerのframespacing属性の両方を0に指定することによって、枠は完全に表示されなくなります。
この方法は、HTML4.0で正式に定義されているものではなく、あくまでユーザーが多い前述の両ブラウザでのみ有効な方法です。この機能を利用することによって、フレームのサイズが変更できなくなるなどの弊害もあることを覚えておいてください。

Sample

```
<HTML lang="ja">
<HEAD>
<TITLE>フレームを区切る枠を完全に消すサンプル</TITLE>
</HEAD>

<FRAMESET rows="70,*" frameborder="0" border="0" frame
spacing="0">
  <FRAME src="title.html">
  <FRAMESET cols="150,*">
    <FRAME src="menu.html">
    <FRAME src="content.html">
  </FRAMESET>
  <NOFRAMES>
  <BODY>
    ∫
  </BODY>
  </NOFRAMES>
</FRAMESET>

</HTML>
```

注意「フレーム利用時の注意」:P.172参照

コラム

フレームをオフにできるブラウザ

フレームを表示しないように設定することのできるブラウザとしてはOperaが有名ですが、実はInternet Explorerのバージョン4.xでも可能だということはあまり知られていないようです。この設定は、Macintosh版のバージョン4.xでも可能ですので、ソフトをお持ちの方は、ぜひ試してみてください。

ただし、バージョン5.xでは設定できなくなっています。

HTML フレームに未対応のブラウザ向けの内容を入れる

<NOFRAMES>～</NOFRAMES>

```
AUTO                                    フレーム未対応用サンプル
 FRAME  logo
 FRAME  menu
 FRAME  content

                    WWW入門

   * HTML
   * Style Sheet
   * JavaScript
```

Lynxでは、NOFRAMESで指定した内容が表示され、かつ各フレームへの文書へのリンクも付けられる。その場合、FRAME要素のname属性の値でリンクされる。

解説

フレームをサポートしていないブラウザを使っている場合や、フレーム機能をオフにしている場合など、フレームが表現できない場合に表示させる内容を指定します。この要素は、<FRAMESET>～</FRAMESET>の範囲の最初か最後に1つだけ配置してください。

また、<NOFRAMES>～</NOFRAMES>の中にはBODY要素を配置して、内容はその中に記述します。内容としては、「フレームに対応したブラウザで見てください」というようなものではなく、フレーム版の代わりとなる内容そのものや、別途作成したフレーム未対応用ページへのリンクなどを入れるようにしてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
 "http://www.w3.org/TR/REC-html40/frameset.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>フレーム未対応用サンプル</TITLE>
</HEAD>
<FRAMESET rows="70,*">
  <FRAME src="title.html" name="logo">
  <FRAMESET cols="150,*">
    <FRAME src="menu.html" name="menu">
    <FRAME src="content.html" name="content">
  </FRAMESET>
  <NOFRAMES>
  <BODY bgcolor="#FFCC00">
  <H1>WWW入門</H1>
  <UL>
  <LI><A href="ht.html">HTML</A></LI>
  <LI><A href="ss.html">Style Sheet</A></LI>
  <LI><A href="js.html">JavaScript</A></LI>
  </UL>
  </BODY>
  </NOFRAMES>
</FRAMESET>
</HTML>
```

リンク先を表示するフレームを指定する

<A href="*URL*" target="フレーム名">〜</A>

Internet Explorer

北のおさかな図鑑

- ソイ
- アブラコ…
- カジカ
- ホッケ
- カレイ
- ヒラメ
- サケ
- マス
- コマイ
- ハゼ
- ウグイ
- イカ

このページでは、北海道で釣ることのできる魚を紹介しています。左のフレームの中から好きな魚の名前をクリックすると写真と解説が表示されます。

Netscape Navigator

北のおさかな図鑑

このページでは、北海道で釣ることのできる魚を紹介しています。左のフレームの中から好きな魚の名前をクリックすると写真と解説が表示されます。

↓

北のおさかな図鑑

ソイ

ソイは夏枯れと呼ばれる時期も含めて春から秋にかけて(場所によっては真冬でも)比較的簡単に釣れるため、北海道のソルトウォーター・フィッシングでは最もベーシックなターゲットといえる。小樽・札幌近郊の釣り場では、クロソイ・ムラソイ(地方名:ハチガラ)・キツネメバル・タヌキメバル・シマゾイなどを釣ることができる。

解説

フレーム内の文書で指定されているリンクは、そのままだとリンク先を同じフレーム内に表示します。

これを他のフレームに表示するようにしたい場合には、target属性を利用します。target属性には、あらかじめFRAME要素のname属性で指定してあるフレームの名前をそのまま指定します。このtarget属性は、フレームとウインドウを同様に扱いますので、リンク先を別のウインドウに表示させることも可能です。

また、フレーム名には「_top」のように「_」で始まる4つの特別な名前があります。これについては、Tip「ターゲット名について」(P.179)を参照してください。

NN2.0
NN3.0
NN4.0

IE3.0
IE4.0
IE5.0

HTML

Sample

・フレームを定義している文書

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
 "http://www.w3.org/TR/REC-html40/frameset.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>北のおさかな図鑑</TITLE>
</HEAD>
<FRAMESET rows="70,*">
  <FRAME src="title.html" name="logo" scrolling="no">
  <FRAMESET cols="150,*">
    <FRAME src="menu.html" name="menu">
    <FRAME src="content.html" name="content">
  </FRAMESET>
  <NOFRAMES>
  <BODY>
   ～
  </BODY>
  </NOFRAMES>
</FRAMESET>
</HTML>
```

・左側のフレームの文書(menu.html)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>北のおさかなメニュー</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<UL>
<LI><A href="soi.html" target="content">ソイ</A>
<LI><A href="ainame.html" target="content">アイナメ</A>
<LI><A href="kajika.html" target="content">カジカ</A>
<LI><A href="hokke.html" target="content">ホッケ</A>
<LI><A href="karei.html" target="content">カレイ</A>
<LI><A href="hirame.html" target="content">ヒラメ</A>
<LI><A href="sake.html" target="content">サケ</A>
<LI><A href="masu.html" target="content">マス</A>
<LI><A href="komai.html" target="content">コマイ</A>
<LI><A href="haze.html" target="content">ハゼ</A>
<LI><A href="ugui.html" target="content">ウグイ</A>
<LI><A href="ika.html" target="content">イカ</A>
</UL>
</BODY>
</HTML>
```

・右側のフレームの文書(content.html)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>右側のフレーム</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<P>
このページでは、北海道で釣ることのできる魚を紹介しています。左のフレームの中から好きな魚の名前をクリックすると写真と解説が表示されます。
</P>
</BODY>
</HTML>
```

「ページの基礎となる内容」の「パスの基準となるURLを指定する」:P.40参照
「リンク」の「リンク先を別のウインドウに表示する」:P.82参照

Tip

ターゲット名について

リンク先を表示するフレームやウインドウをtarget属性で指定する場合、あらかじめ決められている4種類の特別な名前があります。それぞれの名前と機能は次の通りです。

_top	フレームを解除して、リンク先をウインドウ全体に表示する
_parent	リンク先をリンク元のフレームの親フレームに表示する。親フレームがない場合には、「_self」と同様の結果になる
_self	リンク先をリンク元と同じフレームに表示する
_blank	新しいウインドウを開いて、リンク先をそのウインドウに表示する。そのウインドウは、名前のない状態になる

インラインフレームを配置する

```
<IFRAME src="URL" name="フレーム名">～</IFRAME>
```

【その他に指定可能な属性】

width	フレームの幅(ピクセルまたは%)
height	フレームの高さ(ピクセルまたは%)
marginwidth	フレーム内の左右のマージン(ピクセル)
marginheight	フレーム内の上下のマージン(ピクセル)
scrolling	スクロール(auto：自動／yes：あり／no：なし)
frameborder	フレーム枠の表示／非表示(1：表示／0：非表示)
align	行揃え(left・right・center)

Internet Explorer

支援技術

Webアクセシビリティの範囲で利用されるソフトウェアによる支援技術としては、スクリーンリーダーや画面拡大ソフトウェア、スピーチシンセサイザ、グラフィカルなデスクトップのブラウザで利用可能な音声入力ソフトウェアなどがあります。ハードウェアによる支援技術としては、代替キーボードや様々な種類のポインティングデバイスなどがあります。

【用語解説】

左の用語をクリックすると、このフレームに解説が表示されます。

支援技術

Webアクセシビリティの範囲で利用されるソフトウェアによる支援技術としては、スクリーンリーダーや画面拡大ソフトウェア、スピーチシンセサイザ、グラフィカルなデスクトップのブラウザで利用可能な音声入力ソフトウェアなどがあります。ハードウェアによる支援技術としては、代替キーボードや様々な種類のポインティングデバイスなどがあります。

スクリーンリーダー

画面上に表示されている内容を音声で読み上げるソフトウェア。スクリ

Netscape Navigator

支援技術

Webアクセシビリティの範囲で利用されるソフトウェアによる支援技術としては、スクリーンリーダーや画面拡大ソフトウェア、スピーチシンセサイザ、グラフィカルなデスクトップのブラウザで利用可能な音声入力ソフトウェアなどがあります。ハードウェアによる支援技術としては、代替キーボードや様々な種類のポインティングデバイスなどがあります。

解説

ウインドウを分割する形式のフレームではなく、ウインドウの中に独立して表示されるインラインフレームを配置します。フレーム内には、src属性で指定された内容が表示されます。

<IFRAME>～</IFRAME>の間には、このフレームをサポートしていないブラウザや、フレームを表示しないように設定している場合に表示させたい内容を指定しておきます。このフレームは、FRAMESET要素で定義されるフレームとは異なり、<BODY>～</BODY>の範囲で使用され、<!DOCTYPE>に「HTML4.0 Frameset DTD」を指定する必要もありません。

IE3.0 IE4.0 IE5.0

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>インラインフレーム・サンプル</TITLE>
</HEAD>
<BODY bgcolor="white">
<H1>支援技術</H1>
<P>
<IFRAME src="kaisetsu.html" name="term" width="200" height
="120" align="right">
</IFRAME>
Webアクセシビリティの範囲で利用されるソフトウェアによる支援技術としては、<A
href="sr.html" target="term">スクリーンリーダー</A>や<A href
="sm.html" target="term">画面拡大ソフトウェア</A>、<A href="ss.
html" target="term">スピーチシンセサイザ</A>、グラフィカルなデスクトップ
のブラウザで利用可能な<A href="vi.html" target="term">音声入力ソフト
ウェア</A>などがあります。ハードウェアによる支援技術としては、<A href="ak.
html" target="term">代替キーボード</A>や様々な種類の<A href="pd.
html" target="term">ポインティングデバイス</A>などがあります。
<BR clear="right">
</P>
</BODY>
</HTML>
```

HTML

HTMLにスクリプトを組み込む

```
<SCRIPT type="MIMEタイプ">～</SCRIPT>
<SCRIPT type="MIMEタイプ" language="言語名" src="URL">～
</SCRIPT>
```

MIMEタイプ	スクリプト言語のMIMEタイプ(例：text/javascript)
言語名	スクリプト言語の名前(例：JavaScript1.2)
URL	スクリプトを記述した別ファイルのURL

解説

HTML文書内にスクリプトを記述する場合に使用し、スクリプト言語はこの要素の範囲内に記述します。
この時、この要素に未対応のブラウザがスクリプト部分を画面に表示してしまうことを避けるために、通常はスクリプト全体をHTMLでのコメントにしておきます。
この要素は、<HEAD>～</HEAD>と<BODY>～</BODY>の範囲内の任意の位置に置くことができます。type属性はHTML4.0から採用された属性で、仕様上は必ず指定することになっています。language属性は、古くから利用されている属性で、バージョンに依存したスクリプトを使用したい場合などに利用されていますが、この属性は廃止される予定となっています。src属性は、スクリプトを別ファイルに記述しておいて、それを呼び出して使用したい場合に使用します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>スクリプト・サンプル</TITLE>
<SCRIPT type="text/javascript">
<!--
if (self != top)
  top.location.href = self.location.href;
// -->
</SCRIPT>
</HEAD>
<BODY>

</BODY>
</HTML>
```

「ページの基礎となる内容」の「文字セットやスタイルシート・スクリプトの言語を指定する」:P35参照
「JavaScriptについて」の「JavaScriptの記述方法(<SCRIPT>タグの使い方)」:P254参照
「JavaScriptについて」の「JavaScriptの記述方法(JavaScriptの外部呼び込み方法)」:P257参照

HTML スクリプトが実行されない環境用の内容を入れる

<NOSCRIPT>〜</NOSCRIPT>

解説

スクリプトが実行できない場合に、代わりに表示させる内容を指定します。
したがって、スクリプトが実行可能な状況では、ここに記入した内容は表示されません。
この要素は、<BODY>〜</BODY>の範囲内に配置するようにしてください。
また、この要素の内容としては、「スクリプト対応のブラウザで見てください」というものではなく、スクリプトが利用できる場合と同じような意味を持つ内容(または、そのようなページへのリンク)を入れるようにしてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>スクリプトが実行できない環境のためのサンプル</TITLE>
</HEAD>
<BODY>
   ⟆
<NOSCRIPT>
<P>
スクリプトが利用できない環境の方は、
<A href="acs.html">アクセシブル版</A>
をご利用ください。
</P>
</NOSCRIPT>
   ⟆
</BODY>
</HTML>
```

「JavaScriptについて」の「JavaScriptの記述方法(<NOSCRIPT>タグの使い方)」:P256参照

スタイルシートについて

1.スタイルシートについて

HTMLは基本的に文書の構造を示すために使用しますが、それに対して見栄えやレイアウトなどの表示に関する指定を専門に行うのが、このスタイルシートです。
スタイルシートはまだ完全にはサポートされておらず、思い通りに表示されない場合も多いとは思いますが、それでもHTMLだけでは不可能な多くの表現を可能にしてくれます。そして、何よりも大事なことは、スタイルシートを使うことによって、HTMLから表示に関する指定を取り除くことができるということです。
つまり、スタイルシートを適切に使うということは、最終的に「その文書をより多くの人が利用できるようにする」ということにもなります。

実際には、スタイルシートといっても何種類かの言語が存在するのですが、本書でスタイルシートという場合には、現在、最も広く利用が可能と思われるCSS(Cascading Style Sheet)のことを指しています。そして本書では、その中から現在、実際に利用可能なものを中心に解説をしています。

2.基本的な書き方

CSSは、基本的に次のような書式で記述されます。

```
セレクタ { プロパティ:値 }
```

セレクタとは、どの要素に対してスタイルを適用させるかを指定する部分です。この部分でスタイルの適用対象を示し、それに続く{}の中にスタイルを記述します。
プロパティとは、セレクタで指定した要素に適用するスタイルの種類を示す部分です。色を表す「color」や、フォントサイズを表す「font-size」などがこれにあたります。これに続けて「:」記号と値を記述することでスタイルを設定することができます。スタイルは、「;」で区切って複数指定することができます。

・H1要素のフォントサイズを24ポイントに設定した例

```
H1 { font-size: 24pt }
```

・H1要素のフォントサイズを24ポイントに、色を青に設定した例

```
H1 { font-size: 24pt; color: blue }
```

指定するスタイルが多い場合には、次のような書き方もできます。この場合、スタイルとスタイルの間を「;」で区切ることを忘れないようにしてください。

```
H1 {
  font-size: 24pt;
  color: blue;
    ⋮
  }
```

3.スタイルを適用する対象の指定

スタイルを適用させる要素を特定するセレクタには、様々な指定方法があります。
以下に基本的な指定方法を示します。これらは組み合わせて指定することも可能です。

要素名

セレクタとして要素名を指定すると、その要素に対してスタイルを適用します。下の例では、H1要素のフォントサイズを24ポイントに設定しています。

```
H1 { font-size: 24pt }
```

要素名, 要素名, 要素名, ・・・

異なる要素に対して同じスタイルを指定する場合には、要素名を「,」で区切って指定することができます。下の例では、H1要素・H2要素・H3要素の色を青に設定しています。

```
H1, H2, H3 { color: blue }
```

要素名.クラス名
.クラス名

特定のクラスを指定した要素に対してのみ、スタイルを適用します。クラス名の前に要素名を指定した場合は、その要素の中で指定されたクラスが指定されているものだけが対象となります。下の例では、class属性に「myclass」が指定されている要素の色を青に設定しています。

```
.myclass { color: blue }
```

要素名#ID
#ID

特定のIDを指定した要素に対してのみ、スタイルを適用します。IDの前に要素名を指定した場合は、その要素の中で指定されたIDが指定されているものだけが対象となります。下の例では、id属性に「myid」が指定されている要素の色を青に設定しています。

```
#myid { color: blue }
```

要素名 要素名 要素名 ・・・

特定の要素に含まれる要素に対してのみ、スタイルを適用します。上位の要素(階層)から順に、スペースで区切って示していきます。下の例では、DIV要素に含まれているBLOCKQUOTE要素の中にあるP要素の色を青に設定しています。

```
DIV BLOCKQUOTE P { color: blue }
```

A:link
A:visited
A:active
A:hover

リンク部分の状態に応じたスタイルを設定します。標準の状態(link)、すでにページを読み込んだ状態(visited)、リンク部分でマウスのボタンを押している間の状態(active)、マウスカーソルがリンク部分の上にある状態(hover)の各スタイルを設定できます。下の例では、リンク部分の4つの状態について、それぞれ色を設定しています。

```
A:link { color: blue }
A:visited { color: green }
A:active { color: black }
A:hover { color: yellow }
```

4.単位について

CSSで大きさを指定する場合には、数値に次の単位を付けて示すことができます。

これらは絶対的な値を示す単位と、相対的な値を示す単位との2つに分類することができます。

具体的な大きさについては、巻末付録の「Webサイズチャート」を参照してください。

・絶対的な値を示す単位

in	インチ(1インチ=2.54cm)
cm	センチメートル
mm	ミリメートル
pt	ポイント(1ポイント=1/72インチ)
pc	パイカ(1パイカ=12ポイント)

・相対的な値を示す単位

em	その範囲で有効なフォントの高さを1とする単位
ex	その範囲で有効なフォントの小文字の「x」の高さを1とする単位
px	1ピクセルを1とする単位
%	他の基準となる大きさに対する割合(基準はそれぞれ異なります)

5.色の指定方法

CSSで色を指定するには、以下の5つの方法があります。

具体的な色については、巻末付録の「カラーチャート1〜3」を参照してください。

・16進数で指定する -1- (#RRGGBB形式)

HTMLの場合と同様の指定方法です。「#」記号に続けて、RGB(Red・Green・Blue)の各値を2桁ずつの16進数で示します。

例えば、赤を指定する場合には「#FF0000」となります。

・16進数で指定する -2- (#RGB形式)

この指定方法は、上記の方法に似ていますが、RGBの各値をそれぞれ1桁ずつの16進数で示します。この値は、指定したRGBの各1桁の値を2つ続けて並べた数値として解釈されて表示されます。

例えば、「#F36」と指定した場合には「#FF3366」を指定したのと同じ結果になります。したがって、赤を指定する場合には「#F00」となります。

・10進数で指定する(rgb(n,n,n)形式)

この指定方法では、rgbに続く()の中にRとGとBの10進数の値をそれぞれ「,」で区切って示します。RGBの各値は0～255の範囲で指定することができます。

例えば、赤を指定する場合には「rgb(255,0,0)」となります。

・%で指定する(rgb(n%,n%,n%)形式)

この指定方法では、rgbに続く()の中にRとGとBのパーセントの値をそれぞれ「,」で区切って示します。RGBの各値は0%～100%の範囲で指定することができます。

例えば、赤を指定する場合には「rgb(100%,0%,0%)」となります。

・色の名前で指定する

HTML4.0で定義されている16色を名前で直接指定することができます。

また、システムで利用されている色を指定できるキーワードも用意されているのですが、現在のところあまりサポートされていないようなので、ここでは説明を省略します。

6.コメントの書き方

CSSでは、「/* ～ */」で囲った範囲がコメントになります。

コメントの中に更にコメントを入れることはできません。

また、CSSのコメントとしては機能しませんが、古いブラウザなどでスタイルシート部分が表示されてしまうことを避けるために、STYLE要素内でHTMLのコメント記号「<!-- ～ -->」を使用することができるようになっています。

```
<STYLE type="text/css">
  <!--
  .warning { color: #FF0000 }       /* 注意は赤で!    */
  #special { font-weight: bold }    /* ここだけは太字 */
  -->
</STYLE>
```

HTML

NN4.0 IE3.0 IE4.0 IE5.0

HTMLにスタイルシートを組み込む

<STYLE type="text/css">〜</STYLE>

Internet Explorer

スタイルシート・サンプル

この例では、文字色・背景色・左マージンを設定しています。

Netscape Navigator

スタイルシート・サンプル

この例では、文字色・背景色・左マージンを設定しています。

解説

HTMLファイルの中にスタイルを書き込む場合に使用します。
この要素は、必ず<HEAD>〜</HEAD>の範囲内に配置するようにしてください。
<STYLE>〜</STYLE>の範囲には、スタイルを直接書き込みます。この時、スタイルシートに未対応のブラウザがスタイルとして書いた文字自体を表示してしまうことを避けるために、全体をHTMLでのコメントとして「<!-- 〜 -->」で囲うようにします。
この方法は、複数のページで共通するスタイルではなく、主にページ固有のスタイルを記述する場合に使用されます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>スタイルシート・サンプル</TITLE>
<STYLE type="text/css">
  <!--
  BODY {                             /* BODYに適用 */
    color: #330000;                  /* 文字色 */
    background-color: #FFCC00;       /* 背景色 */
    margin-left: 3em                 /* 左マージン */
  }
  -->
</STYLE>
</HEAD>
<BODY>
<H1>スタイルシート・サンプル</H1>
<P>
この例では、文字色・背景色・左マージンを設定しています。
</P>
</BODY>
</HTML>
```

NN4.0
IE3.0
IE4.0
IE5.0

HTML

別ファイルのスタイルシートを読み込む

```
<LINK rel="stylesheet" href="URL" type="text/css">
```

URL	スタイルを記述したファイルのURL

Internet Explorer

スタイルシート・サンプル

この例では、別ファイルで定義したスタイルを読み込んで、
文字色・背景色・左マージンを設定しています。

Netscape Navigator

スタイルシート・サンプル

この例では、別ファイルで定義したスタイルを読み込んで、
文字色・背景色・左マージンを設定しています。

解説

スタイルのみを記述した別のファイル(拡張子は.css)を読み込んで利用する場合に使用します。
この要素は、必ず<HEAD>〜</HEAD>の範囲内に配置するようにしてください。
複数のファイルを読み込みたい場合には、この要素を必要な数だけ配置することができます。この方法を使用すると、1つのスタイルシートを複数のHTMLファイルから利用することができますので、同じサイト内で統一させたいスタイルを書いておくと便利です。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>スタイルシート・サンプル</TITLE>
<LINK rel="stylesheet" href="default.css" type="text/css">
</HEAD>
```

```
<BODY>
<H1>スタイルシート・サンプル</H1>
<P>
この例では、別ファイルで定義したスタイルを読み込んで、
文字色・背景色・左マージンを設定しています。
</P>
</BODY>
</HTML>
```

【スタイルを記述したファイル(default.css)の内容】

```
BODY {                              /* BODYに適用 */
  color: #330000;                   /* 文字色 */
  background-color: #FFCC00;        /* 背景色 */
  margin-left: 3em                  /* 左マージン */
}
```

「ページの基礎となる内容」の「他のページとの関係を表す」:P.38参照
Tip「ページ同士の関係を表す値」:P.39参照

重要

スタイル競合時の優先順位

スタイルを記述した別ファイルを続けて複数読み込んだ場合、その中で指定されているスタイルが競合する可能性が出てきます。その場合には、より後に読み込んだスタイルが有効になります。

また、これとは別にSTYLE要素やstyle属性を利用したスタイルが、更に競合する可能性もあります。この場合は、より細かい部分で指定しているスタイルが優先されます。

例えば、セレクタとして要素を指定しているものよりはクラスで指定しているもの、クラスで指定しているものよりはIDで指定しているものが優先されます。style属性を利用して要素に直接指定しているスタイルは、更に優先順位が高くなります。

HTML

タグ内にスタイルシートを組み込むための属性

<要素名 style="スタイル">

Internet Explorer

スタイルシート・サンプル

この例では、BODY要素に文字色と背景色を、H1要素に文字色を、P要素には左マージンを設定しています。

Netscape Navigator

スタイルシート・サンプル

この例では、BODY要素に文字色と背景色を、H1要素に文字色を、P要素には左マージンを設定しています。

解説

任意のタグにstyle属性を追加して、そこだけに有効なスタイルを書き込むことができます。複数のスタイルを指定したい場合には、スタイルを「;」で区切って指定します。この方法を利用する場合は、そこに書かれているスタイルシート言語(CSS以外もあり得ます)を特定する手段がありませんので、META要素を利用して必ずデフォルトのスタイルシート言語を宣言しておくようにしてください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>スタイルシート・サンプル</TITLE>
<META http-equiv="Content-Style-Type" content="text/css">
</HEAD>
<BODY style="color: #330000; background-color: #FFCC00">
<H1 style="color: #FFFFFF">スタイルシート・サンプル</H1>
<P style="margin-left: 3em">
この例では、BODY要素に文字色と背景色を、H1要素に文字色を、
P要素には左マージンを設定しています。
</P>
</BODY>
</HTML>
```

「ページの基礎となる内容」の「文字セットやスタイルシート・スクリプトの言語を指定する」:P.35参照

HTML

定義済みのスタイルを適用させるための属性

```
#ID名 {スタイル}        ←<要素名 id="ID名">
.クラス名 {スタイル}    ←<要素名 class="クラス名">
```

Internet Explorer	Netscape Navigator
これは普通の段落です。 【注意1】○○に注意してください。 これは普通の段落です。 【注意2】○○に注意してください。 **これは特別な段落です。**	これは普通の段落です。 【注意1】○○に注意してください。 これは普通の段落です。 【注意2】○○に注意してください。 **これは特別な段落です。**

解説

あらかじめID名やクラス名を指定してスタイルを定義しておき、それをid属性やclass属性で指定した要素だけに適用させることができます。

id属性で指定するID名は、そのHTMLファイルの中で1つの要素にしか適用することができませんので、例外的な固有の部分に対して使用します。class属性で指定するクラス名は、同じものを複数の要素に対して指定することができます。

「#ID名」を「要素名#ID名」のように書くと、指定した要素に対してのみ有効なID名を定義をすることができます。同様に「.クラス名」を「要素名.クラス名」のように書くと、指定した要素に対してのみ有効なクラス名が定義できます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>定義済みのスタイルを適用するサンプル</TITLE>
<STYLE type="text/css">
  <!--
  .warning { color: #FF0000 }        /* 注意は赤で!     */
  #special { font-weight: bold }     /* ここだけは太字 */
  -->
</STYLE>
</HEAD>
<BODY>
<P>これは普通の段落です。</P>
<P class="warning">【注意1】○○に注意してください。</P>
<P>これは普通の段落です。</P>
<P class="warning">【注意2】○○に注意してください。</P>
<P id="special">これは特別な段落です。</P>
</BODY>
</HTML>
```

HTML

任意の範囲にスタイルを適用させるためのタグ

<SPAN>～</SPAN>
<DIV>～</DIV>

Internet Explorer

CENTER要素を使うなら…

CENTER要素を使うなら、
このDIV要素を利用して
スタイルシートのtext-alignを
使った方がよいでしょう。

Netscape Navigator

CENTER要素を使うなら…

CENTER要素を使うなら、
このDIV要素を利用して
スタイルシートのtext-alignを
使った方がよいでしょう。

解説

SPAN要素はそれがインラインの要素であることを、DIV要素はそれがブロックレベルの要素であることを示し、他に要素としての意味を特別に持ってはいません。したがって、他の要素で特に指定されていない範囲を、スタイルを適用する目的の要素として指定する場合などに利用します。
多くの場合、id属性やclass属性などと共に利用されます。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>任意の範囲にスタイルを適用させるサンプル</TITLE>
<STYLE type="text/css">
  <!--
   DIV.special { text-align: center }    /* センタリング */
  .strict { color: red }                 /* 赤で表示 */
  .deprecated { color: green }           /* 緑で表示 */
  -->
</STYLE>
</HEAD>
<BODY>
<DIV class="special">
<H1>CENTER要素を使うなら…</H1>
<P>
<SPAN class="deprecated">CENTER要素</SPAN>を使うなら、<BR>
この<SPAN class="strict">DIV要素</SPAN>を利用して<BR>
スタイルシートのtext-alignを<BR>
使った方がよいでしょう。
</P>
</DIV>
</BODY>
</HTML>
```

HTML

文字色を指定する

color: 色

NN4.0

IE3.0

IE4.0

IE5.0

Internet Explorer

文字色の指定

スタイルシートで文字色を指定するのは意外と簡単です。実際には、色の指定方法は色々とありますが、ここでは基本的な方法を紹介します。

Netscape Navigator

文字色の指定

スタイルシートで文字色を指定するのは意外と簡単です。実際には、色の指定方法は色々とありますが、ここでは基本的な方法を紹介します。

解説

文字の色(前景色)を設定します。
仕様上は、すべての要素に対して設定できることになっていますが、ブラウザのサポート状況によっては色が変化しないものもあります。

Sample

【CSS】

```
BODY {
  color: #330000;              /* 文字色 */
  background-color: #FF6600; /* 背景色 */
}
A:link { color: #FFFF00 }     /* リンク */
A:visited { color: #FFFF66 } /* すでに見たリンク */
A:active { color: #000000 }  /* クリックした時のリンク */
A:hover { color: #FFFFFF }   /* カーソルが上にある時のリンク */
H1 { color: #FFFFFF }        /* レベル1の見出し */
```

【HTML】

```
<BODY>
<H1>文字色の指定</H1>
<P>
スタイルシートで文字色を指定するのは意外と簡単です。実際には、
<A href="colorspec.html">色の指定方法</A>は色々とありますが、
ここでは<A href="color01.html">基本的な方法</A>を紹介します。
</P>
</BODY>
```

「スタイルシートについて」の「スタイルを適用する対象の指定」:P.185参照

「スタイルシートについて」の「色の指定方法」:P.187参照

HTML

背景色を指定する

background-color: 色

Internet Explorer

Netscape Navigator

背景色の指定

スタイルシートを利用すると、すべての要素に対して背景色を設定することができます。

名前	年齢	性別
岡蔵 龍一	35	男

解説

背景の色を設定します。
色の値として「transparent」を指定すると、背景が透明になって下の背景と同じ色になります。ここで設定した色はマージンと枠の部分には適用されません。
仕様上は、すべての要素に対して設定できることになっていますが、ブラウザのサポート状況によっては色が変化しないものもあります。

Sample

【CSS】

```
BODY { background-color: #FF6600 }   /* 全体の背景はオレンジ */
H1 { background-color: white }       /* 見出しの背景は白 */
P { background-color: #FFFF00 }      /* 段落の背景は黄色 */
TABLE { background-color: white }    /* 表の背景は白 */
```

【HTML】

```
<BODY>
<H1>背景色の指定</H1>
<P>
スタイルシートを利用すると、すべての要素に対して背景色を設定する
ことができます。
</P>
<TABLE border="2">
<TR><TH>名前</TH><TH>年齢</TH><TH>性別</TH></TR>
<TR><TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD></TR>
</TABLE>
</BODY>
```

HTML

NN4.0 IE4.0 IE5.0

背景画像を指定する

background-image: url("*URL*")

URL	画像のURL

Internet Explorer

背景画像の指定

スタイルシートを利用すると、すべての要素に対して背景画像を設定することができます。

名前	年齢	性別
岡蔵　龍一	35	男

Netscape Navigator

背景画像の指定

スタイルシートを利用すると、すべての要素に対して背景画像を設定することができます。

名前	年齢	性別
岡蔵　龍一	35	男

解説

背景の画像を設定します。
仕様上は、すべての要素に対して設定できることになっていますが、ブラウザのサポート状況によっては背景画像が表示されないものもあります。

Sample

【CSS】
```
BODY { background-image: url("back.gif") }
H1, P, TABLE { background-image: url("cloth.gif") }
```

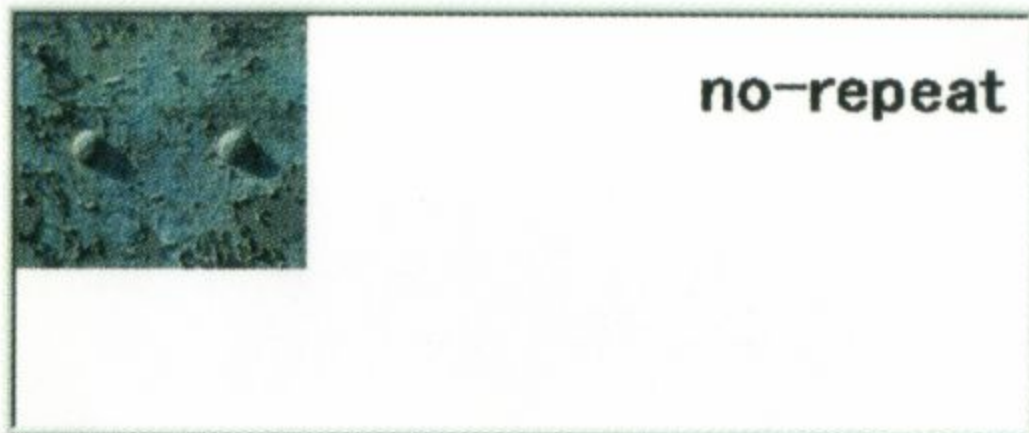
【HTML】
```
<BODY>
<H1>背景画像の指定</H1>
<P>
スタイルシートを利用すると、すべての要素に対して背景画像を
設定することができます。
</P>
<TABLE border="2">
<TR><TH>名前</TH><TH>年齢</TH><TH>性別</TH></TR>
<TR><TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD></TR>
</TABLE>
</BODY>
```

HTML 背景画像の並び方を指定する

background-repeat: *並び方*

【並び方】

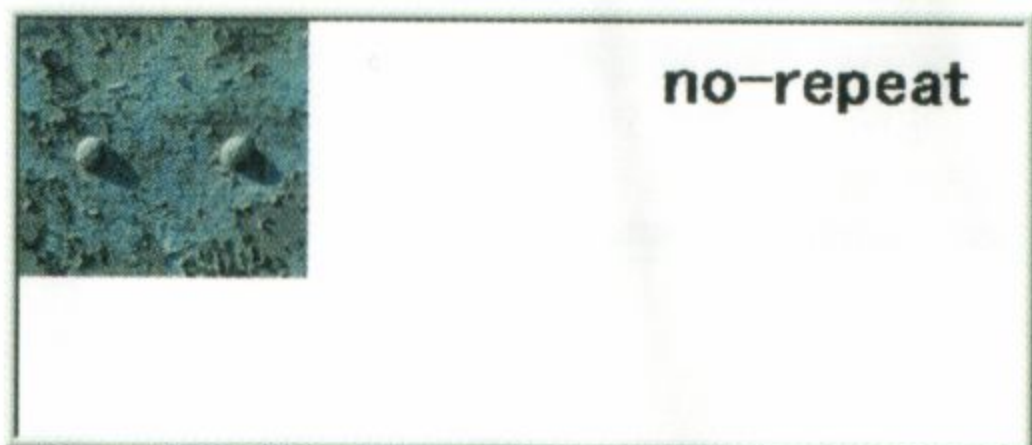
repeat	縦横にタイル状に繰り返して表示
repeat-x	左上から横方向にのみ繰り返して表示
repeat-y	左上から縦方向にのみ繰り返して表示
no-repeat	繰り返さずに1つだけ表示

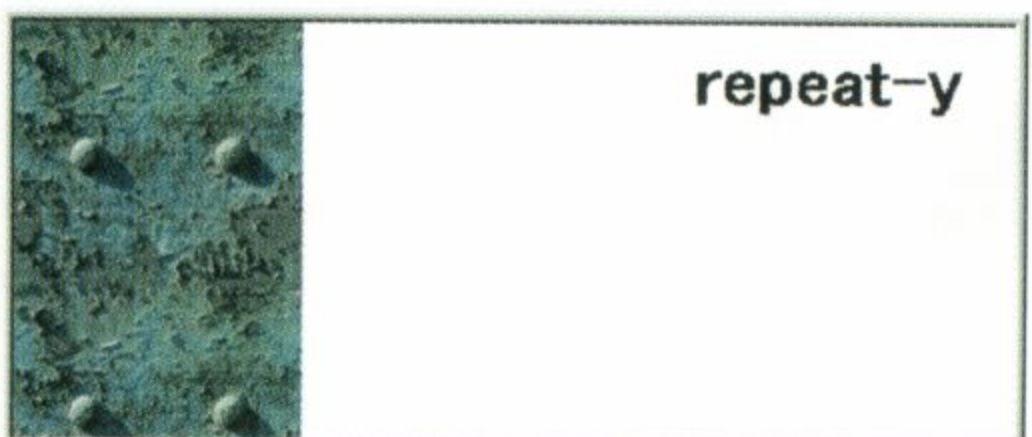
Internet Explorer

no-repeat

repeat-x

repeat-y

repeat

Netscape Navigator

no-repeat

repeat-x

repeat-y

repeat

解説

背景画像が指定された場合の、画像の並び方を設定します。
仕様上は、すべての要素に対して設定できることになっていますが、ブラウザのサポート状況によってはうまく表示されない場合もあります。

Sample

【CSS】

```
BODY {
  background-color: white;
  background-image: url("back.gif");
  background-repeat: no-repeat
}
H1 { text-align: right }
```

【HTML】

```
<BODY>
<H1>no-repeat</H1>
</BODY>
```

NN4.0 IE4.0 IE5.0

背景画像の表示位置を指定する

background-position: *表示位置*

IE4.0
IE5.0

【表示位置】

横位置	縦位置	単位付きの数値または%
横位置	縦位置	left・right・center・top・bottom

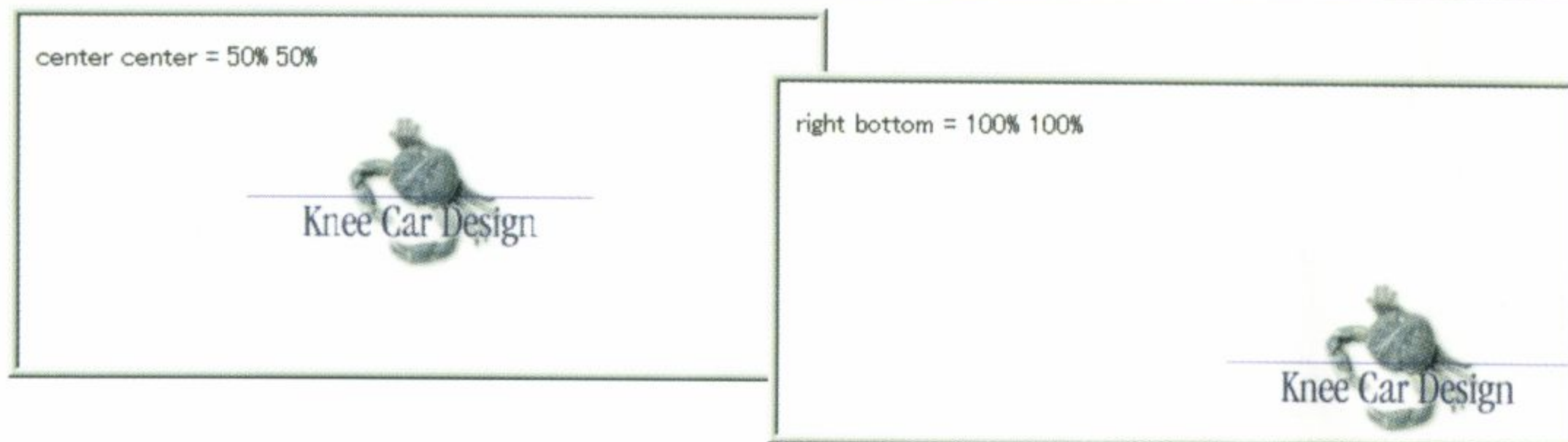
Internet Explorer

center center = 50% 50%

Knee Car Design

right bottom = 100% 100%

Knee Car Design

解説

背景画像が指定された場合の、画像の表示位置を設定します。
単位付きの数値と%による指定の場合は、横位置・縦位置の順にスペースで区切って指定します。値が1つしか指定されなかった場合は、横位置が指定されたことになり、その場合の縦位置は「50%」の位置になります。単位付きの数値の場合は、領域の左上から画像の左上までの距離を指定します。%による指定の場合は、領域の指定した割合の位置に、画像の同じ割合の位置が合わせて表示されます。
これらは混在させて指定することが可能です。位置を示すキーワード(left、topなど)は、それぞれleftとtopは「0%」、centerは「50%」、rightとbottomは「100%」を指定した場合と同じ結果になります。これらは順不同で指定することができ、1つしか指定しなかった場合は、もう一方がcenterになります。

Sample

【CSS】
```
BODY {
  background-color: white;
  background-image: url("logo.gif");
  background-repeat: no-repeat;
  background-position: right bottom
}
```

【HTML】
```
<BODY>
<P>right bottom = 100% 100%</P>
</BODY>
```

背景画像を固定する

background-attachment: *固定するかどうか*

【固定するかどうか】

fixed	背景画像の位置を固定する
scroll	背景画像を他の内容と共にスクロールさせる

IE4.0

IE5.0

Internet Explorer

ヒラメはフィッシュイーター

ヒラメを初めて釣りあげた時、

その鋭い歯に驚くアングラーも多い。

その主食は、生きた魚類、エビ類、イカ類などで、

動いているものを食べる習性がある。

ヒラメを狙う場合は、より遠くへキャストすることも

重要だが、ルアーはよりリアルなものをチョイスしたい。

私の経験では、メタルジグよりはミノー、ミノーよりは

ソフトルアーがより好釣果を招くようだ。

解説 背景画像が指定された場合に、それを他の内容と共にスクロールさせるか、他の内容とは別に固定された位置で表示するかを指定します。初期値は「scroll」です。
仕様上は、すべての要素に対して設定できることになっていますが、ブラウザのサポート状況によってはうまく固定されない場合もあります。

Sample

【CSS】
```
BODY {
  background-color: white;
  background-image: url("hirame.gif");
  background-repeat: no-repeat;
  background-attachment: fixed;
  margin-top: 100px;                /* 上マージン */
  text-align: right                 /* 右揃え */
}
P { line-height: 2em }              /* 行の高さ */
```

【HTML】
```
<BODY>

<H1>ヒラメはフィッシュイーター</H1>
<P>
ヒラメを初めて釣りあげた時、<BR>
その鋭い歯に驚くアングラーも多い。<BR>
その主食は、生きた魚類、エビ類、イカ類などで、<BR>
動いているものを食べる習性がある。
</P>

<P>
ヒラメを狙う場合は、より遠くへキャストすることも<BR>
重要だが、ルアーはよりリアルなものをチョイスしたい。<BR>
私の経験では、メタルジグよりはミノー、ミノーよりは<BR>
ソフトルアーがより好釣果を招くようだ。
</P>

</BODY>
```

IE4.0 IE5.0

Tip

背景をまとめて設定する

背景に関する指定は、「background」というプロパティを使用すると、まとめて行うことができます。この場合、背景に関する各プロパティで利用できる値を、必要なだけスペースで区切って任意の順序で指定できます。

【例】 `P { background: url("back.gif") #FFFFFF 50% no-repeat fixed }`

HTML

行の高さを指定する

line-height： *行の高さ*

NN4.0 IE3.0 IE4.0 IE5.0

行の高さ	normal・数値・単位付きの数値・%

Internet Explorer

line-height: normal

行の高さは文章の読みやすさに影響します。
狭すぎても広すぎても読みにくくなります。
行数が少ない場合は行間が少なくても比較
的読みやすく、1行の文字数が多い場合は
行間が狭いとかなり読みにくくなります。

line-height: 150%

行の高さは文章の読みやすさに影響します。
狭すぎても広すぎても読みにくくなります。
行数が少ない場合は行間が少なくても比較
的読みやすく、1行の文字数が多い場合は
行間が狭いとかなり読みにくくなります。

Netscape Navigator

line-height: normal

行の高さは文章の読みやすさに影響します。
狭すぎても広すぎても読みにくくなります。
行数が少ない場合は行間が少なくても比較
的読みやすく、1行の文字数が多い場合は
行間が狭いとかなり読みにくくなります。

line-height: 150%

行の高さは文章の読みやすさに影響します。
狭すぎても広すぎても読みにくくなります。
行数が少ない場合は行間が少なくても比較
的読みやすく、1行の文字数が多い場合は
行間が狭いとかなり読みにくくなります。

解説

行の高さを設定します。
単位を付けないで数値だけを指定した場合は、フォントサイズにその値を掛けた高さに設定されます。%による指定も、フォントサイズに対する割合になります。
マイナスの値は指定できないことになっていますので、注意してください。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>行の高さサンプル</TITLE>
</HEAD>
<BODY>
```

```
<H1>line-height: normal</H1>
<P style="line-height: normal">
行の高さは文章の読みやすさに影響します。<BR>
狭すぎても広すぎても読みにくくなります。<BR>
行数が少ない場合は行間が少なくても比較<BR>
的読みやすく、1行の文字数が多い場合は<BR>
行間が狭いとかなり読みにくくなります。
</P>

<H1>line-height: 150%</H1>
<P style="line-height: 150%">
行の高さは文章の読みやすさに影響します。<BR>
狭すぎても広すぎても読みにくくなります。<BR>
行数が少ない場合は行間が少なくても比較<BR>
的読みやすく、1行の文字数が多い場合は<BR>
行間が狭いとかなり読みにくくなります。
</P>

</BODY>
</HTML>
```

Tip「フォントをまとめて設定する」:P.216参照

コラム

Netscape Navigator4.x のレイヤーについて・1

Netscape Navigator4.xでは、独自のタグを利用してレイヤーを作成することができます。レイヤーは、任意の位置に配置して、重ねたり消したりクリッピングすることなどができます。<LAYER>~</LAYER>で囲った範囲の内容が、レイヤーとして表示されます。属性としては、次のものが指定できます。

・**id="レイヤー名"**
他のレイヤーやJavaScriptから、このレイヤーを参照する場合に利用する名前を指定します。初期のバージョンでは、この属性の代わりに「name」属性を使用していました(現在でも利用できます)。

・**src="内容のURL"**
レイヤーの内容を別のHTMLファイルにしておく場合に利用します。内容としてJava Scriptを含むこともできます。

・**pagex="左からの位置" pagey="上からの位置"**
レイヤーを配置する位置をピクセルで指定します。ウインドウの描画領域の左上からレイヤーの左上までの距離を指定します。

この続きはP.206へ!

マージンを設定する

HTML

```
margin-top: マージン         ←上マージン
margin-bottom: マージン      ←下マージン
margin-left: マージン        ←左マージン
margin-right: マージン       ←右マージン
margin: マージン             ←上・右・下・左マージン
```

マージン	単位付きの数値・%

Internet Explorer

この文書全体には、左右のマージンとして80ピクセル、上下のマージンとして0ピクセルが設定されています。

この段落には、更に上下左右に40ピクセルのマージンが設定されています。したがって、文書全体のマージンと合わせると左右には120ピクセルのマージンがあることになります。

Netscape Navigator

この文書全体には、左右のマージンとして80ピクセル、上下のマージンとして0ピクセルが設定されています。

この段落には、更に上下左右に40ピクセルのマージンが設定されています。したがって、文書全体のマージンと合わせると左右には120ピクセルのマージンがあることになります。

解説

マージンを設定します。
「margin」を利用すると、上下左右のマージンを1度に設定することができます。その場合、値をスペースで区切って指定しますが、あたえられた値の個数によって、次のようにマージンが設定されます。

・値が1つの場合　値1→上下左右
・値が2つの場合　値1→上下　値2→左右
・値が3つの場合　値1→上　値2→左右　値3→下
・値が4つの場合　値1→上　値2→右　値3→下　値4→左

各マージンの初期値は0です。
仕様上は、すべての要素に対して設定できることになっていますが、ブラウザのサポート状況によってはうまく設定されない場合もあります。

Sample

NN4.0
IE3.0
IE4.0
IE5.0

HTML

【CSS】

```
BODY { margin: 0px 80px }    /* 上下0pixel、左右80pixel */
P.special { margin: 40px }   /* 上下左右40pixel */
```

【HTML】

```
<BODY>

<P>
この文書全体には、左右のマージンとして80ピクセル、上下の
マージンとして0ピクセルが設定されています。
</P>

<P class="special">
この段落には、更に上下左右に40ピクセルのマージンが設定
されています。したがって、文書全体のマージンと合わせると
左右には120ピクセルのマージンがあることになります。
</P>

</BODY>
```

コラム

Netscape Navigator4.xのレイヤーについて・2

・left="左からの位置" top="上からの位置"

レイヤーを配置する位置をピクセルで指定します。親レイヤーがある場合はその左上から、ない場合はウインドウの描画領域の左上からレイヤーの左上までの距離を指定します。

・z-index="重なる順序"

レイヤーの重なる順序を正の整数で指定します。この値が大きいものほど上に表示されます。

・above="上のレイヤー名" below="下のレイヤー名"

above属性を指定した場合は、「上のレイヤー名」で指定したレイヤーのすぐ下に、このレイヤーが表示されます。below属性を指定した場合は、「下のレイヤー名」で指定したレイヤーのすぐ上に、このレイヤーが表示されます。

このコラムはP.204から続いています。(続きはP.209へ!!)

HTML

行揃えを指定する

text-align: *行揃え位置*

NN4.0

IE3.0

IE4.0

IE5.0

行揃え位置　left · right · center

Internet Explorer

夜明けの行揃え

実をいうと、行揃えには「left・right・center」の他に「justify」というものがある。いわゆる「両端揃え」のことである。しかし、色々試してはみたのだが、これが実際には・・・

【文:岡蔵　龍一】

- ご感想をおよせください -

(C) Copyright 1999 OKARYU,Inc. All Rights Reserved.

Netscape Navigator

夜明けの行揃え

実をいうと、行揃えには「left・right・center」の他に「justify」というものがある。いわゆる「両端揃え」のことである。しかし、色々試してはみたのだが、これが実際には・・・

【文:岡蔵　龍一】

- ご感想をおよせください -

(C) Copyright 1999 OKARYU,Inc. All Rights Reserved.

HTML

解説

行揃えを設定します。
行揃え位置として指定する「left · right · center」は、それぞれ「左揃え・右揃え・中央揃え」を表しています。ブロックレベルの要素に対して指定できます。

Sample

【CSS】

```
H1 { text-align: center }
P.author { text-align: right }
DIV.foot { text-align: center }
```

【HTML】

```
<BODY>

<H1>夜明けの行揃え</H1>
<P>
実をいうと、行揃えには「left·right·center」の他に
「justify」というものがある。いわゆる「両端揃え」のこと
である。しかし、色々試してはみたのだが、これが実際には・・・
</P>
<P class="author">【文:岡蔵　龍一】</P>

<DIV class="foot">
<P>- ご感想をおよせください -</P>
<P>(C) Copyright 1999 OKARYU,Inc. All Rights Reserved.</P>
</DIV>

</BODY>
```

スタイルシートを使う

HTML

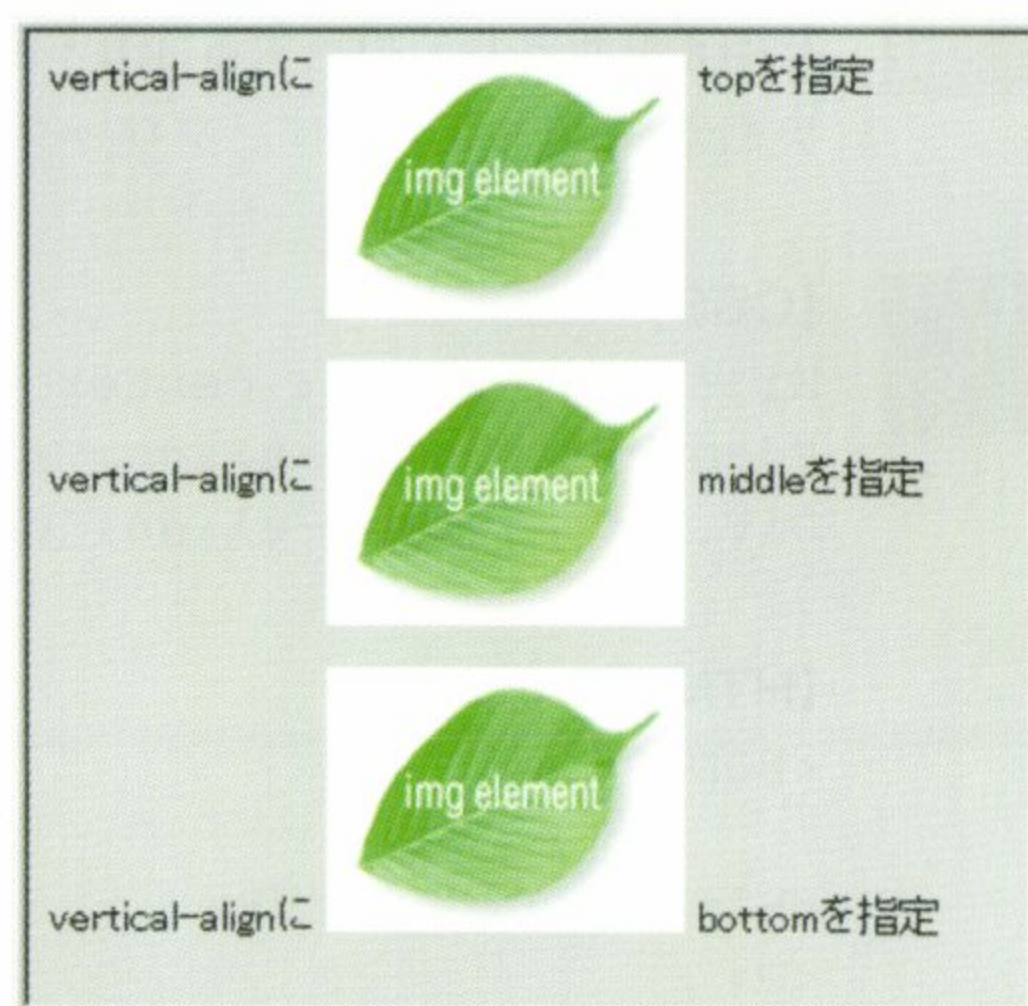
縦方向の位置関係を指定する

vertical-align: *行揃え位置*

【行揃え位置】

top	上に揃える
middle	中心に揃える
bottom	下に揃える
baseline	親要素のベースラインに揃える(初期値)
text-top	親要素のフォントの上に揃える
text-bottom	親要素のフォントの下に揃える
super	上付き文字の位置にベースラインを揃える
sub	下付き文字の位置にベースラインを揃える
単位つきの数値	親要素のベースラインを0とした値の＋－
%	親要素のベースラインを0とした行高さの割合

Internet Explorer

Netscape Navigator

解説

インライン要素の縦方向の位置を設定します。
縦方向の位置には様々な指定ができますが、ブラウザのサポート状況によってはうまく表示されないものもあります。

Sample

【CSS】

```
.t1 { vertical-align: top }
.t2 { vertical-align: middle }
.t3 { vertical-align: bottom }
```

【HTML】

```
<BODY>

<P>vertical-alignに
<IMG class="t1" src="leaf.gif" width="140" height="100"
 alt="">
topを指定</P>

<P>vertical-alignに
<IMG class="t2" src="leaf.gif" width="140" height="100"
 alt="">
middleを指定</P>

<P>vertical-alignに
<IMG class="t3" src="leaf.gif" width="140" height="100"
 alt="">
bottomを指定</P>

</BODY>
```

NN4.0

IE4.0
IE5.0

HTML

コラム

Netscape Navigator4.xのレイヤーについて・3

・width="幅" height="高さ"

レイヤーの幅と高さをピクセル、又は%で指定します。

・clip="左,上,右,下"

この属性を指定すると、レイヤーをクリッピングして、指定した範囲だけを表示することができます。「左,上,右,下」は、それぞれピクセル数で指定します。

・visibility="表示／非表示"

「表示／非表示」には、「show」・「hide」・「inherit」が指定できます。「show」はレイヤーを表示、「hide」は非表示、「inherit」はこのレイヤーを含む上位のレイヤーと同様の状態になります。

このコラムはP.206から続いています。（続きはP.211へ!!）

フォントスタイルを指定する

HTML

font-style: *スタイル*
text-decoration: *装飾*

スタイル	normal · italic · oblique
装飾	none · underline · overline · line-through · blink

Internet Explorer

font-style

normal, *italic*, *oblique*

text-decoration

none, underline, overline, line-through

Netscape Navigator

font-style

normal, *italic*, oblique

text-decoration

none, underline, overline, line-through

解説

フォントのスタイルや装飾を設定します。
ここでいう「italic」とは、通常の文字よりも続け書きに近い草書体風にデザインされた斜体のフォントを指します。「oblique」とは、標準のフォントを単純に斜めにしたものです。「underline・overline・line-through」は、それぞれ文字に対して下線・上線・字消し線を付け、「blink」は文字を点滅させます。
スタイルを標準にするためには「normal」を、装飾をなくすためには「none」を指定します。

Sample

【CSS】

```
P { font-family: "Times New Roman", Times, serif }
A:link, A:visited, A:active { text-decoration: none }
.it { font-style: italic }
.ob { font-style: oblique }
.un { text-decoration: underline }
.ov { text-decoration: overline }
.lt { text-decoration: line-through }
```

【HTML】
```
<BODY>

<H1>font-style</H1>
<P>
normal,
<SPAN class="it">italic</SPAN>,
<SPAN class="ob">oblique</SPAN>
</P>

<H1>text-decoration</H1>
<P>
<A href="test.html">none</A>,
<SPAN class="un">underline</SPAN>,
<SPAN class="ov">overline</SPAN>,
<SPAN class="lt">line-through</SPAN>
</P>

</BODY>
```

コラム

Netscape Navigator4.x のレイヤーについて・4

・**bgcolor="背景色"　background="背景画像のURL"**
BODY要素と同様に背景色と背景画像を指定します。これらが指定されない場合には、レイヤーは透明になります。

※実際には、この他にも位置を相対的に指定できる<ILAYER>要素や、レイヤーに対応していないブラウザ用の<NOLAYER>要素もあります。詳細は、次のサイトを参照してください。

Netscape DevEdge Online
http://developer.netscape.com/

このコラムはP.209から続いています。

NN4.0
IE3.0
IE4.0
IE5.0

フォントの太さを指定する

HTML

```
font-weight: 太さ
```

【太さ】

normal・bold・lighter・bolder
100・200・300・400・500・600・700・800・900

Internet Explorer	Netscape Navigator
フォントの太さ 普通の文字とボールドの文字	フォントの太さ 普通の文字とボールドの文字

解説

フォントの太さを設定します。
「bold」を指定すると、一般的な太字になります。同じフォントファミリー中に異なる太さのフォントがある場合、「lighter」は1段階細いフォントに、「bolder」は1段階太いフォントに設定されます。
また、太さは100～900の数値で指定することもできます。この場合、標準の太さ「normal」は400で、「bold」は700に相当します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD>
<TITLE>フォントの太さサンプル</TITLE>
</HEAD>
<BODY>
<H1>フォントの太さ</H1>
<P>
普通の文字と<SPAN style="font-weight: bold">ボールドの文字</SPAN>
</P>
</BODY>
</HTML>
```

フォントサイズを指定する

```
font-size: サイズ
```


【サイズ】

単位付きの数値・%・smaller・larger
xx-small・x-small・small・medium・large・x-large・xx-large

Internet Explorer

フォントのサイズ

xx-small, x-small, small, medium, large, x-large, xx-large

smaller, 12pt, 14pt, 18px, 180%

Netscape Navigator

フォントのサイズ

xx-small, x-small, small, medium, large, x-large, xx-large

smaller, 12pt, 14pt, 18px, 180%

解説

フォントのサイズを設定します。
%で指定した場合は、指定した要素の親である要素のフォントサイズに対する割合になります。
また、キーワードを利用することによって、「xx-small」から「xx-large」までの7段階のサイズを表現することができます。この場合、「medium」が標準のサイズです。
「smaller」と「larger」は、親要素のフォントサイズに対して、それぞれ1段階小さいサイズと1段階大きいサイズに設定します。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD>
<TITLE>フォントのサイズ・サンプル</TITLE>
</HEAD>
<BODY>
```

```
<H1>フォントのサイズ</H1>
<P>
<SPAN style="font-size: xx-small">xx-small</SPAN>,
<SPAN style="font-size: x-small">x-small</SPAN>,
<SPAN style="font-size: small">small</SPAN>,
<SPAN style="font-size: medium">medium</SPAN>,
<SPAN style="font-size: large">large</SPAN>,
<SPAN style="font-size: x-large">x-large</SPAN>,
<SPAN style="font-size: xx-large">xx-large</SPAN>
</P>

<P>
<SPAN style="font-size: smaller">smaller</SPAN>,
<SPAN style="font-size: 12pt">12pt</SPAN>,
<SPAN style="font-size: 14pt">14pt</SPAN>,
<SPAN style="font-size: 18px">18px</SPAN>,
<SPAN style="font-size: 180%">180%</SPAN>
</P>

</BODY>
</HTML>
```

注意

CSS1とCSS2のフォントサイズの違い

フォントサイズは、smallやmediumなどのキーワードで7段階のサイズを指定することができますが、実はこの7段階の大きさの比率がCSS1とCSS2では異なっています。

CSS1の時には、各段階の比率として1.5倍が推奨されていましたが、CSS2になってこの比率が1.2倍に変更されました。つまり、そのブラウザが対応しているCSSのバージョンによって、キーワードで指定した場合のフォントサイズが異なるということです。

フォントサイズをキーワードで指定する場合には、十分ご注意ください。

HTML

フォントの種類を指定する

font-family: フォント名, フォント名, フォント名, ・・・

【フォント名】

フォントファミリー名
serif・sans-serif・cursive・fantasy・monospace

Internet Explorer

ここはゴシック体で、ここは
明朝体にしてみます。

Netscape Navigator

ここはゴシック体で、ここは
明朝体にしてみます。

解説

フォントの種類を設定します。
フォント名は1つでも指定できますが、カンマで区切って複数指定することができます。その場合は、より先(左)に指定されているフォントで、ユーザーの環境で表示可能なものが採用されます。
フォント名の中にスペースが含まれている場合は、必ずその前後を「"」または「'」で囲うようにしてください。また、フォント名として、フォントの種類を表すキーワードを指定することもできます。各キーワードは、それぞれ次のような種類を表しています。

・serif	明朝系 (例：Times New Roman, ＭＳ Ｐ明朝)
・sans-serif	ゴシック系 (例：Helvetica, Arial, ＭＳ Ｐゴシック)
・cursive	草書(筆記)体系 (例：Caflisch Script, Adobe Poetica)
・fantasy	装飾的なフォント (例：Critter, Cottonwood)
・monospace	等幅フォント (例：Courier New, ＭＳ ゴシック)

これらは、指定したすべてのフォント名が有効でない場合の最終的な指定として、常に指定しておいた方がよいでしょう。

Sample

【CSS】

```
P { font-size: 24pt }
.gthc { font-family: "ＭＳ Ｐゴシック",Osaka,sans-serif }
.mnch { font-family: "ＭＳ Ｐ明朝",細明朝体,serif }
```

【HTML】

```
<BODY>
<P>
<SPAN class="gthc">ここはゴシック体で、</SPAN>
<SPAN class="mnch">ここは明朝体にしてみます。</SPAN>
</P>
</BODY>
```

Tip

フォントをまとめて設定する

フォントに関する指定は、「font」というプロパティを使用すると、まとめて行うことができます。この場合、フォントに関する各プロパティで利用できる値を、次の順にスペースで区切って指定します。

1.font-styleで指定できる値
2.font-weightで指定できる値
3.font-sizeで指定できる値(省略不可)
4.「/」+line-heightで指定できる値
5.font-familyで指定できる値(省略不可)

【例】 `P { font: italic bold 12pt/15pt serif }`

HTML

文字間隔・単語間隔を指定する

letter-spacing: *文字間隔*
word-spacing: *単語間隔*

IE4.0
IE5.0

文字間隔・単語間隔	normal・単位付きの数値

Internet Explorer

letter-spacing

letter-spacing: normal
l e t t e r - s p a c i n g :   0 . 5 e m

文字間隔:標準
文 字 間 隔 : 1 e m

word-spacing

It specifies spacing behavior between words.
It specifies spacing behavior between words.

Internet Explorer (Mac4.5)

letter-spacing

letter-spacing: normal
l e t t e r - s p a c i n g :   0 . 5 e m

文字間隔：標準
文 字 間 隔 ： 1 e m

word-spacing

It specifies spacing behavior between words.
It   specifies   spacing   behavior   between   words.

解説 letter-spacingは文字と文字の間隔を、word-spacingは単語と単語の間隔を設定します。数値を指定した場合は、標準の値に対してプラス(マイナスも可)されます。

Sample

【CSS】

```
.ls1 { letter-spacing: 0.5em }
.ls2 { letter-spacing: 1em }
.ws { word-spacing: 0.8em }
```

【HTML】

```
<BODY>

<H1>letter-spacing</H1>
<P>
letter-spacing: normal<BR>
<SPAN class="ls1">letter-spacing: 0.5em</SPAN>
</P>
```

HTML

スタイルシートを使う

```
<P>
文字間隔：標準<BR>
<SPAN class="ls2">文字間隔：1em</SPAN>
</P>

<H1>word-spacing</H1>
<P>
It specifies spacing behavior between words.<BR>
<SPAN class="ws">It specifies spacing behavior between words.</SPAN>
</P>

</BODY>
```

HTML

幅と高さを指定する

width: *幅*
height: *高さ*

幅・高さ　単位付きの数値・%

Internet Explorer

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

Netscape Navigator

名前	年齢	性別
岡蔵　龍一	35	男
伊藤　清治	29	男

解説

内容を表示する部分の幅と高さを設定します。
内容を表示する部分とは、枠やマージン、枠と内容の間の領域を除いた、内容を表示可能な領域のことです。ブロックレベルの要素と置き換え要素(IMG・INPUT・TEXTAREA・SELECTなど)、横列を除くテーブル関連要素に対して設定できます。
%で指定する場合は、親要素に対する割合になります。

Sample

【CSS】

```
TH.name {
  width: 120px;
  height: 50px
}
```

【HTML】

```
<BODY>
<TABLE border="2">
<TR>
<TH class="name">名前</TH>
<TH>年齢</TH><TH>性別</TH>
</TR>
<TR>
<TD>岡蔵　龍一</TD><TD>35</TD><TD>男</TD>
</TR>
<TR>
<TD>伊藤　清治</TD><TD>29</TD><TD>男</TD>
</TR>
</TABLE>
</BODY>
```

HTML

枠の太さを指定する

```
border-top-width: 太さ          ←上の枠の太さ
border-bottom-width: 太さ       ←下の枠の太さ
border-left-width: 太さ         ←左の枠の太さ
border-right-width: 太さ        ←右の枠の太さ
border-width: 太さ              ←上・右・下・左の枠の太さ
```

太さ	単位付きの数値・thin・medium・thick

Internet Explorer

枠の太さ

img element

Netscape Navigator

枠の太さ

img element

解説

枠の太さを設定します。
枠の太さは、すべての要素に対して設定可能です。
「border-width」を利用すると、上下左右の枠の太さを1度に設定することができます。その場合、値をスペースで区切って指定しますが、あたえられた値の個数によって、次のように枠の太さが設定されます。

・値が1つの場合　値1→上下左右
・値が2つの場合　値1→上下　値2→左右
・値が3つの場合　値1→上　値2→左右　値3→下
・値が4つの場合　値1→上　値2→右　値3→下　値4→左

thin・medium・thickは、それぞれ細い枠・中くらいの枠・太い枠に設定します。
この場合の実際の太さはブラウザによって異なります。

【CSS】

```
H1 {
  border-style: solid;
  border-width: 1px 0px 1px 10px
}
IMG {
  border-style: solid;
  border-width: 2px
}
```

【HTML】

```
<BODY>
<H1>枠の太さ</H1>
<P>
<IMG src="leaf.gif" width="140" height="100" alt="">
</P>
</BODY>
```

Tip

Netscape Navigatorでは画像の枠は設定できない

左ページのサンプル画像を見てお気づきの方も多いと思いますが、Netscape Navigator 4.xでは画像に対して枠の設定をすることはできません(おかしな部分に枠が表示されます)。したがって、現在では画像の枠をスタイルシートで消すこともできません。Netscape Navigatorのユーザーも考慮すると、しばらくの間はIMG要素のborder属性を利用するのも仕方がないことかもしれません。

NN4.0
IE4.0
IE5.0

HTML

枠の色を指定する

```
border-top-color: 色        ←上の枠の色
border-bottom-color: 色     ←下の枠の色
border-left-color: 色       ←左の枠の色
border-right-color: 色      ←右の枠の色
border-color: 色            ←上・右・下・左の枠の色
```

Internet Explorer

枠の色を設定する

スタイルシートでは、枠に関するさまざまな設定をすることができます。HTMLだけでは、このようなことは簡単にはできませんね。

Netscape Navigator

枠の色を設定する

スタイルシートでは、枠に関するさまざまな設定をすることができます。HTMLだけでは、このようなことは簡単にはできませんね。

解説

枠の色を設定します。
枠の色は、すべての要素に対して設定可能です。
「border-color」を利用すると、上下左右の枠の色を1度に設定することができます。
その場合、値をスペースで区切って指定しますが、あたえられた値の個数によって、次のように枠の色が設定されます。

- 値が1つの場合　値1→上下左右
- 値が2つの場合　値1→上下　値2→左右
- 値が3つの場合　値1→上　値2→左右　値3→下
- 値が4つの場合　値1→上　値2→右　値3→下　値4→左

色の値として「transparent」を設定することもできます。その場合、枠の色は透明になります。
なお、この値の初期値は「color: 色」で設定されている値となります。

Sample

【CSS】

```
BODY { background-color: white }
H1 {
  padding-left: 10px;
  border-style: solid;
  border-width: 2px 2px 2px 15px;   /* 左の枠のみ太く */
  border-color: #009933             /* 色はグリーン系 */
}
P {
  padding: 10px 15px;
  border-style: solid;
  border-width: 0px 0px 1px;   /* 下の枠のみ1pixelで表示 */
  border-color: #CCCCCC        /* 色は明るいグレー */
}
```

【HTML】

```
<BODY>
<H1>枠の色を設定する</H1>
<P>
スタイルシートでは、枠に関するさまざまな設定をすることができます。
HTMLだけでは、このようなことは簡単にはできませんね。
</P>
</BODY>
```


Tip

枠をまとめて設定する

枠に関する指定は、「border」というプロパティ、または「border-top」・「border-bottom」・「border-left」・「border-right」の各プロパティを使用すると、まとめて行うことができます。この場合、枠に関する各プロパティで利用できる値を、必要なだけスペースで区切って任意の順序で指定できます。

【例】 `P { border: solid 2px #CCCCCC }`

HTML NN4.0 IE4.0 IE5.0

枠の形式を指定する

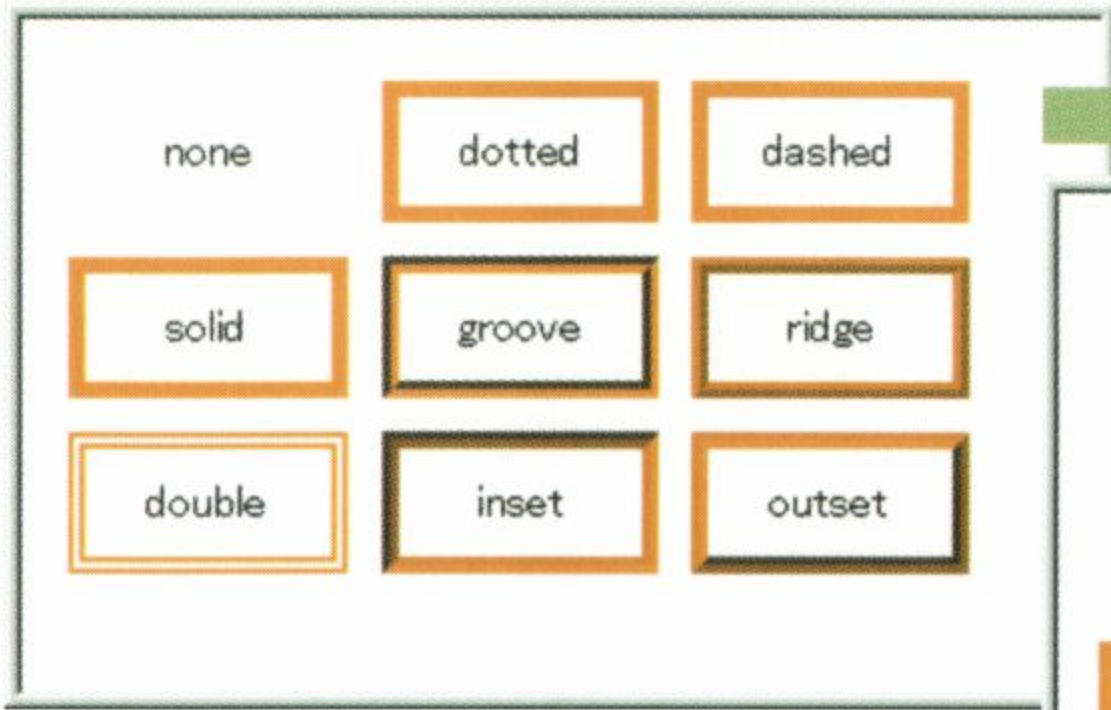
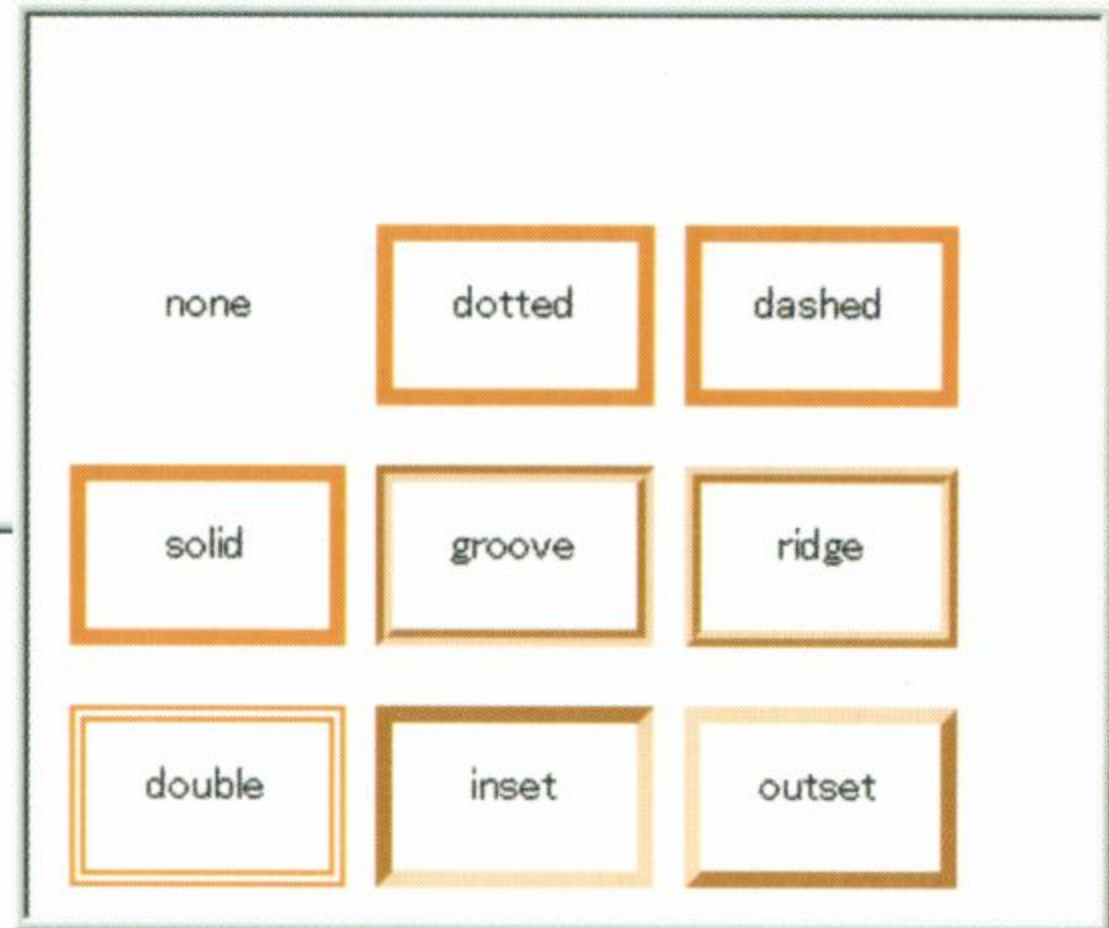
```
border-top-style: 形式        ←上の枠の形式
border-bottom-style: 形式     ←下の枠の形式
border-left-style: 形式       ←左の枠の形式
border-right-style: 形式      ←右の枠の形式
border-style: 形式            ←上・右・下・左の枠の形式
```

形式	none · hidden · dotted · dashed · solid · double · groove · ridge · inset · outset

Internet Explorer

none　dotted　dashed
solid　groove　ridge
double　inset　outset

Netscape Navigator

none　dotted　dashed
solid　groove　ridge
double　inset　outset

解説

枠の形式を設定します。
枠の形式は、すべての要素に対して設定可能です。
「border-style」を利用すると、上下左右の枠の形式を1度に設定することができます。その場合、値をスペースで区切って指定しますが、あたえられた値の個数によって、次のように枠の形式が設定されます。

・値が1つの場合　値1→上下左右
・値が2つの場合　値1→上下　値2→左右
・値が3つの場合　値1→上　値2→左右　値3→下
・値が4つの場合　値1→上　値2→右　値3→下　値4→左

「none」と「hidden」はどちらも枠を表示せず、枠の太さも0に設定します。ただし、テーブルのセルの枠として重なりあった場合には、「none」は他の値を優先し、「hidden」は自分自身の値を優先します。この値の初期値は「none」です。

※現時点では「dotted」と「dashed」は、あまりサポートされていないようです。

Sample

【CSS】

```
BODY { background-color: white }
TD { padding: 5px }
P {
  width: 100px;
  padding: 10px;
  text-align: center;
  border-width: 6px;
  border-color: #FF6600
}
```

【HTML】

```
<BODY>
<TABLE>
<TR>
<TD><P style="border-style: none">none</P></TD>
<TD><P style="border-style: dotted">dotted</P></TD>
<TD><P style="border-style: dashed">dashed</P></TD>
</TR>
<TR>
<TD><P style="border-style: solid">solid</P></TD>
<TD><P style="border-style: groove">groove</P></TD>
<TD><P style="border-style: ridge">ridge</P></TD>
</TR>
<TR>
<TD><P style="border-style: double">double</P></TD>
<TD><P style="border-style: inset">inset</P></TD>
<TD><P style="border-style: outset">outset</P></TD>
</TR>
</TABLE>
</BODY>
```

Tip「枠の形式の種類」:P.227参照

NN4.0
IE4.0
IE5.0

HTML

枠と内容の間の空間を設定する

```
padding-top: 幅          ←上の空間
padding-bottom: 幅       ←下の空間
padding-left: 幅         ←左の空間
padding-right: 幅        ←右の空間
padding: 幅              ←上・右・下・左の空間
```

幅	単位付きの数値・%

Internet Explorer

普通の段落に枠をつけたもの

paddingを20pixelに設定したもの

Netscape Navigator

普通の段落に枠をつけたもの

paddingを20pixelに設定したもの

解説

枠と内容を表示する領域の間の空間の幅を設定します。
この幅は、すべての要素に対して設定可能です。
「padding」を利用すると、上下左右の幅を1度に設定することができます。その場合、値をスペースで区切って指定しますが、あたえられた値の個数によって、次のように幅が設定されます。

・値が1つの場合　値1→上下左右
・値が2つの場合　値1→上下　値2→左右
・値が3つの場合　値1→上　値2→左右　値3→下
・値が4つの場合　値1→上　値2→右　値3→下　値4→左

Sample

【CSS】
```
BODY { background-color: white }
P {
  border-style: solid;
  border-color: #FF6600;
  border-width: 2px
}
P.readable { padding: 20px }
```

【HTML】
```
<BODY>
<P>普通の段落に枠をつけたもの</P>
<P class="readable">paddingを20pixelに設定したもの</P>
</BODY>
```

Tip

枠の形式の種類

スタイルシートで設定できる枠の形式は、以下の通りです。

none	枠の値を表示しない。枠の太さも0に設定され、テーブルのセルの枠同士が重なった場合は、他の値が優先される
hidden	枠の値を表示しない。枠の太さも0に設定されるが、テーブルのセルの枠同士が重なった場合は、この値が優先される
dotted	枠を点で表示する
dashed	枠を点線で表示する
solid	枠を線で表示する
double	枠を2本線で表示する
groove	枠が引っ込んでいるように立体的に表示する
ridge	枠が出っ張っているように立体的に表示する
inset	ボックス全体が引っ込んでいるように立体的に表示する
outset	ボックス全体が出っ張っているように立体的に表示する

リンク部分のスタイルを指定する

HTML

```
A:link { スタイル }          ←普通のリンク
A:visited { スタイル }       ←すでに見たリンク
A:active { スタイル }        ←クリックした時のリンク
A:hover { スタイル }         ←カーソルが上に乗った時のリンク
```

Internet Explorer

リンクに関連するスタイル

スタイルシートを利用すると、一部のリンクだけスタイルや色などを変えることができます。リンクにスタイルを適用する場合は「疑似クラス」というものを利用します。

[次ページ] [トップ] [前ページ]

Netscape Navigator

リンクに関連するスタイル

スタイルシートを利用すると、一部のリンクだけスタイルや色などを変えることができます。リンクにスタイルを適用する場合は「疑似クラス」というものを利用します。

[次ページ] [トップ] [前ページ]

解説

リンク部分のスタイルを設定します。
クラスの指定と組み合わせることによって、任意の部分に対して別々のスタイルを適用することもできます。
特定のクラスを指定したA要素のスタイルを設定したい場合は、「A.クラス名:link」のように指定します。また、特定のクラスを指定したある要素に含まれるA要素のスタイルを設定したい場合は、「.クラス名 A:link」のように指定します。

Sample

【CSS】

```
BODY {
  color: black;                 /* 文字色 */
  background-color: white       /* 背景色 */
}
```

```
/* リンク部分の色の設定 */
A:link { color: blue }          /* 普通のリンク色 */
A:visited { color: navy }       /* すでに見たリンク色 */
A:active { color: red }         /* クリックした時のリンク色 */
A:hover { color: aqua }         /* カーソルが乗った時のリンク色 */

/* リンク部分の下線を消す */
A:link, A:visited, A:active, A:hover { text-decoration:
 none }

/* 個別にクラスを指定したリンクだけ色を変える */
A.special { font-weight: bold }           /* 太字 */
A.special:link { color: #006699 }         /* 渋い青 */
A.special:visited { color: #003366 }      /* 渋い紺色 */
A.special:active { color: #990000 }       /* 渋い赤 */
A.special:hover { color: #3399CC }        /* 渋い水色 */

/* 特定の範囲のリンクだけ色を変える */
.navbar {
  text-align: center;
  border-top: solid #999999 1px;
  padding-top: 10px
}
.navbar A:link { color: #FF3300 }     /* オレンジ */
.navbar A:visited { color: #FF9900 }  /* 薄いオレンジ */
.navbar A:active { color: #FFFF00 }   /* 黄色 */
.navbar A:hover { color: #FF0000 }    /* 赤 */
```

【HTML】

```
<BODY>

<H1>リンクに関連するスタイル</H1>
<P>
<A href="css.html">スタイルシート</A>を利用すると、1部の
<A href="link.html">リンク</A>だけスタイルや色などを変える
ことができます。リンクにスタイルを適用する場合は「<A href=
"pseudo.html" class="special">疑似クラス</A>」という
ものを利用します。
</P>

<P class="navbar">
[<A href="next.html">次ページ</A>]
[<A href="top.html">トップ</A>]
[<A href="prev.html">前ページ</A>]
</P>

</BODY>
```

HTML 左右への配置と回り込みを指定する

float: *配置位置*

配置位置	left・right・none

Internet Explorer

スタイルシートを利用すると、この例のように画像や表などにテキストを回り込ませることができます。回り込んだテキストが収まりきらない場合には、テキストは画像の下に表示されます。この回り込みを解除するためには「clear」プロパティを使用します。

Netscape Navigator

スタイルシートを利用すると、この例のように画像や表などにテキストを回り込ませることができます。回り込んだテキストが収まりきらない場合には、テキストは画像の下に表示されます。この回り込みを解除するためには「clear」プロパティを使用します。

解説

指定した要素を左または右に配置して、その反対側に他の要素を回り込ませます。leftは指定した要素を左に、rightは右に配置します。noneを指定すると左右への配置と回り込みは行いません。
回り込みを指定した後にそれを解除するためには、「clear」プロパティを利用します。

Sample

【CSS】

```
BODY { background-color: white }
IMG { float: left }
```

【HTML】

```
<BODY>
<P>
<IMG src="fish.gif" width="230" height="145" alt="サンプル画像">
スタイルシートを利用すると、この例のように画像や表などにテキストを回り込ませることができます。回り込んだテキストが収まりきらない場合には、テキストは画像の下に表示されます。この回り込みを解除するためには「clear」プロパティを使用します。
</P>
</BODY>
```

回り込みを解除する

clear: どちら側の要素に対して解除するか

【どちら側の要素に対して解除するか】

left	左側の要素に対する回り込みを解除
right	右側の要素に対する回り込みを解除
both	両側の要素に対する回り込みを解除
none	回り込みを解除しない

Internet Explorer

スタイルシートを利用すると、この例のように画像や表などにテキストを回り込ませることができます。

さて、この回り込みを解除するためには「clear」プロパティを使用するわけですが・・・

Netscape Navigator

スタイルシートを利用すると、この例のように画像や表などにテキストを回り込ませることができます。

さて、この回り込みを解除するためには「clear」プロパティを使用するわけですが・・・

解説

ある要素を左、又は右に配置してテキストなどを回り込ませた場合の、回り込みを解除します。

ブロックレベルの要素に対して指定することができます。

Sample

【CSS】

```
BODY { background-color: white }
IMG { float: left }
.newcont { clear: left }
```

【HTML】
```
<BODY>

<P>
<IMG src="fish.gif" width="230" height="145" alt="サンプル
画像">
スタイルシートを利用すると、この例のように画像や表などにテキスト
を回り込ませることができます。
</P>

<P class="newcont">
さて、この回り込みを解除するためには「clear」プロパティを使用する
わけですが・・・
</P>

</BODY>
```

Tip

リストで使えるマーカーの種類

スタイルシートで設定できるリストのマーカー(マークや数字)の種類は、以下の通りです。

disc	黒まるに設定する
circle	白まるに設定する
square	四角に設定する
decimal	1から始まる十進数の数字に設定する
lower-roman	小文字のローマ字に設定する
upper-roman	大文字のローマ字に設定する
lower-alpha	小文字のアルファベットに設定する
upper-alpha	大文字のアルファベットに設定する
none	マーカーを表示しないように設定する

リストのマークや番号の形式を変える

HTML

list-style-type: *種類*

種類	disc・circle・square・decimal・lower-roman・upper-roman・lower-alpha・upper-alpha・none

Internet Explorer

• disc	○ circle	■ square
• 黒まる	○ 白まる	■ 四角
• 黒まる	○ 白まる	■ 四角
1. decimal	i. lower-roman	I. upper-roman
2. 数字	ii. ローマ数字	II. ローマ数字
3. 十進数	iii. 小文字	III. 大文字
a. lower-alpha	A. upper-alpha	none
b. アルファベット	B. アルファベット	なし
c. 小文字	C. 大文字	なし

Netscape Navigator

• disc	• circle	■ square
• 黒まる	• 白まる	■ 四角
• 黒まる	• 白まる	■ 四角
1. decimal	i. lower-roman	I. upper-roman
2. 数字	ii. ローマ数字	II. ローマ数字
3. 十進数	iii. 小文字	III. 大文字
a. lower-alpha	A. upper-alpha	none
b. アルファベット	B. アルファベット	なし
c. 小文字	C. 大文字	なし

解説

リストのマークや番号の形式を設定します。
仕様上は、他にも指定できる種類があるのですが、ここでは実際に使用できる種類を紹介しています。

Sample

```
<BODY>
<UL style="list-style-type: circle">
<LI>circle</LI>
<LI>白まる</LI>
<LI>白まる</LI>
</UL>
</BODY>
```

カーソルの形を指定する

HTML

cursor: *形状*

IE4.0
IE5.0

形状	auto・crosshair・default・pointer・move・text・wait・help・e-resize,・ne-resize・nw-resize・n-resize・se-resize・sw-resize・s-resize・w-resize

Internet Explorer

以下のソースをコピーしてお使いください。

```
<SCRIPT language="JavaScript" type="text/javascript">
    <!--
    if (self != top) top.location.href = self.location.href;
    // -->
</SCRIPT>
```

Netscape Navigator

以下のソースをコピーしてお使いください。

```
<SCRIPT language="JavaScript" type="text/javascript">
    <!--
    if (self != top) top.location.href = self.location.href;
    // -->
</SCRIPT>
```

解説

マウスなどのポインティングデバイスのカーソルが、その要素の上に乗っている時のカーソルの形状を設定します。
このプロパティは、すべての要素に対して設定することができます。

Sample

【CSS】

```
CODE { cursor: text }
```

【HTML】

```
<BODY>
<P>以下のソースをコピーしてお使いください。</P>
<PRE><CODE>
&lt;SCRIPT language="JavaScript" type="text/javascript"&gt;
    &lt;!--
    if (self != top) top.location.href = self.location.href;
    // --&gt;
&lt;/SCRIPT&gt;
</CODE></PRE>
</BODY>
```

形状名	Win	Mac	説明
auto			ブラウザが状況に応じて自動設定
crosshair	+	+	十字型
default			標準(矢印型)
pointer			リンク部分の上にある時の形状
move			移動可能であることを示す形状
text			テキスト選択用の形状
wait			処理中であることを示す形状
help	?	?	ヘルプを示す形状
n-resize			リサイズ可能：上方向
s-resize			リサイズ可能：下方向
w-resize			リサイズ可能：左方向
e-resize			リサイズ可能：右方向
ne-resize			リサイズ可能：右上方向
nw-resize			リサイズ可能：左上方向
se-resize			リサイズ可能：右下方向
sw-resize			リサイズ可能：左下方向

HTML

印刷時の改ページを指定する

```
page-break-before: always
page-break-after: always
```

解説

印刷時に、指定した部分の前または後で改ページをするように設定します。「page-break-before」は指定した要素の前で、「page-break-after」は指定した要素の後で改ページします。
仕様上は、指定できる値が他にも用意されているのですが、現在利用できるのはこの値だけのようです。
このプロパティは印刷時にのみ有効で、画面表示には影響を与えません。

Sample

```
H1 { page-break-before: always }        /* 見出しの前は改ページ */
TABLE { page-break-before: always }     /* 表の前は改ページ */
```

アクセシビリティについて

アクセシビリティについて説明する前に、Webコンテンツを利用する状況(環境)にはどのようなものがあるかを考えてみましょう。現在、実際にありえる状況として、次のようなものが考えられます。

・一般的なデスクトップ環境で、マウスやキーボードを使うことができる
・マウスやキーボードなどを使うことができない
・小さな画面やモノクロの画面で見ている
・特定の色を識別することができない
・小さな文字を読むのが困難なため、大きく表示させている
・動いたり点滅する文字を読むことが困難
・古いブラウザや一般的ではないブラウザを利用している
・テキスト以外のデータを表示しないように設定している
・テキストしか表示することができない環境からアクセスしている
・内容を音声や点字などに変換して利用している
・その他の様々な状況

アクセシビリティとは、一般に「アクセス性」や「アクセス容易性」などと訳されていますが、簡単にいえば「情報を得ることができて、利用可能にすること」のような意味です。

つまり、上記のどのような状況でもその情報を利用することができるページは、「アクセシビリティが高い」というわけです。そして、そのようなページを「アクセシブルなページ」とも表現します。

では、「アクセシブルなページ」にするためには、具体的にはどうすればよいのでしょうか？

まず言えることは、文書構造とその表示方法を分離するということと、テキストを上手に利用するということです。テキストデータは、音声や点字に変換することができますし、もちろん画面上で見ることもできるアクセシブルなデータ形式です。他の様々な形式のデータの代替データとしてテキストを用意しておくだけで、多くの人がそれを利用できるようになります。

アクセシビリティの詳細については、ガイドラインや専門のサイト(索引「ホームページ作成関連リンク」：P.626)を参照していただくとして、ここでは一般的なアクセシビリティを考慮した場合の注意点をいくつか紹介しておきます。

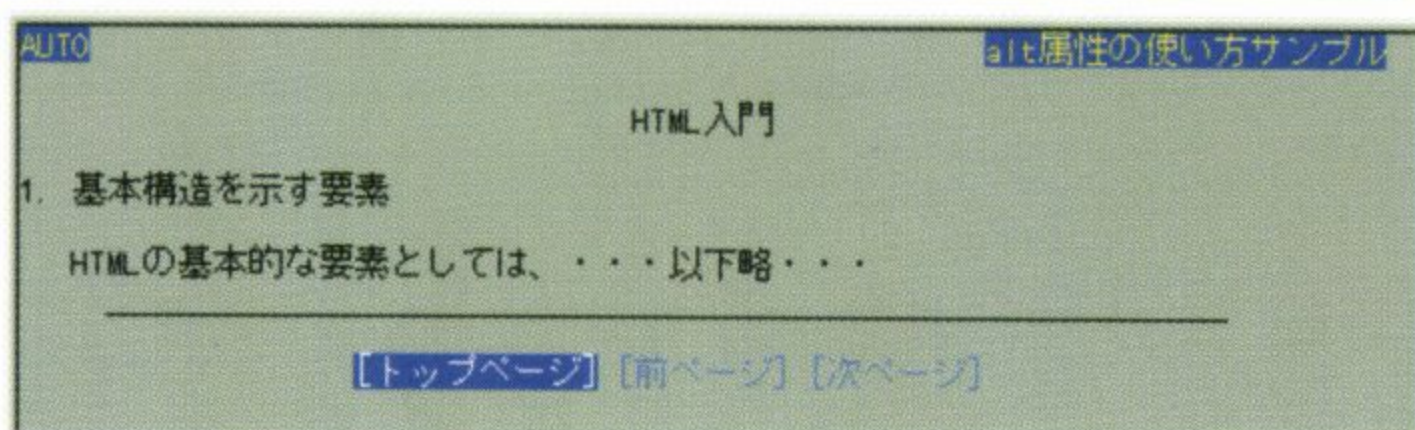
- テキスト以外の形式のデータには、同じ役割や意味のあるテキストデータを付ける(alt属性やリンクなどを利用)
- HTML4.0で廃止予定となっている要素や属性を使用しない
- レイアウトをする目的で、表を使用しない
- 自動的に新しいウインドウを開くことは避ける
- 文字を点滅させたり、動かしたりすることは避ける
- 色の識別が困難な人やモノクロ画面を使用している人でも、判読可能な配色にする
- 一定間隔で内容を自動更新するページを作ることは避ける
- リンク部分の言葉は、その部分を読んだだけでも意味の分かるものにする(リンク先の内容を明示する言葉にする。「Click here」や「ここをクリック」などの言葉は使わない)

HTML

画像の代わりのテキストを指定しておく

<IMG ～ alt="*画像と同等の意味のあるテキスト*">

Internet Explorer

Lynx

解説

alt属性には、画像が表示できない場合に、画像の代わりにそこに配置されるテキストを指定します。
この属性を指定する上で重要なことは、「alt」は「alternative(代わりの)」の意味で、ここには画像の「説明」を書いておくのではなく、あくまで「代わりとなる、画像と同じ意味や役割をもったテキスト」を指定するということです。ツールチップで画像の説明を表示したい場合には、title属性を使用するのが正しい方法です。
alt属性を正しく指定しておくと、画像を表示しないように設定している人やテキストブラウザを利用している人、音声や点字によるアクセスをしている人なども、その内容を理解することができるようになります。しかし、この場合にもいくつか注意する点があります。

・必要がなければ「alt=""」と空文字を指定する
　例えば、文脈上意味がない画像などの場合は、無理にテキストを入れるとかえって混乱する場合があります。

- **連続した複数の画像のalt属性は、連続して表現される**
  複数の画像が続けて配置されている場合には、alt属性で指定したテキストがそのまま続けて表示されてもおかしくならないように、スペースや記号、句読点などをうまく入れるようにしてください。

- **文章中の画像のalt属性は、前後のテキストに続けて表現される**
  この場合も、上の例と同様に句読点などを適切に入れるようにしてください。

他にも注意する点はあるのですが、実際にどうなるかを確認してみるのが1番よい方法です。テキストブラウザで見ておかしな部分がなければ、ほぼ問題はないと考えてよいでしょう。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
<TITLE>alt属性の使い方サンプル</TITLE>
</HEAD>
<BODY bgcolor="white">

<H1>
<IMG src="images/logo.gif" width="234" height="54" alt="
HTML入門">
</H1>
<H2>
<IMG src="images/maru.gif" width="20" height="20" alt="
1.">
基本構造を示す要素
</H2>
<P>HTMLの基本的な要素としては、・・・以下略・・・</P>
<HR>
<P align="center">
<A href="index.html">
<IMG src="images/top.gif" alt="[トップページ]" border="0"></A>
<A href="contents.html">
<IMG src="images/prev.gif" alt=" [前ページ]" border="0"></A>
<A href="section2.html">
<IMG src="images/next.gif" alt=" [次ページ]" border="0"></A>
</P>

</BODY>
</HTML>
```

HTML

タブで移動する順序を指定する

<要素名 〜 tabindex="タブ順">

IE4.0
IE5.0

タブ順	0〜32767までの数字

解説

タブキーを押した場合に移動する項目の順序を設定します。
この属性が指定された要素は、指定された数字の小さなものから順に移動します。ただし、値として0が指定されている要素と、属性自体が指定されていない要素は、1以上の数値が指定されているものの後に移動することになります。複数の要素に同じ値が設定されている場合は、文書の中で先にあるものに先に移動します。
この属性は、次の要素の属性として指定することができます。

A・AREA・BUTTON・INPUT・OBJECT・SELECT・TEXTAREA

タブ順を適切に設定しておくと、マウスが使えない状態(環境)の時に非常に便利です。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
  ∫
</HEAD>
<BODY>
<FORM action="/cgi-bin/formmail.cgi" method="post">
<P>
<LABEL>
お名前：<INPUT type="text" name="namae" tabindex="1">
</LABEL><BR>
<LABEL>
メール：<INPUT type="text" name="email" tabindex="2">
</LABEL>
</P>
<P>
<INPUT type="submit" value="送信" tabindex="3">
<INPUT type="reset" value="クリア" tabindex="4">
</P>
</FORM>
</BODY>
</HTML>
```

ショートカットキーを割り当てる

<要素名　～　accesskey="ショートカットキー">

ショートカットキー	HTML4.0で利用可能な文字セット中の1文字

解説

この属性を指定した要素に対して、ショートカットキーを割り当てます。
利用方法は使用しているOSに依存するのですが、Windowsであれば「alt」キー、Macintoshであれば「command」キーを同時に押して利用します。
この属性は、次の要素の属性として指定することができます。

A・AREA・BUTTON・INPUT・LABEL・LEGEND・TEXTAREA

この機能によって、マウスが使えない状態(環境)でも、特定のリンク部分を選択したりボタンを押したりすることなどが素早く行えるようになります。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
  ～
</HEAD>
<BODY>
  ～
<P>
<A href="../contents.html" accesskey="C">Table of
Contents</A>
</P>
  ～
</BODY>
</HTML>
```

HTML

IE4.0
IE5.0

何語で書かれているかを指定する

<要素名 lang="言語">

【言語】

ja	日本語
en	英語
en-US	アメリカ英語
fr	フランス語
de	ドイツ語
zh	中国語　　など

解説

その要素の内容と属性の値の言語を指定します。
基本的な言語を示すだけでなく、途中で異なる言語が出てくる場合にも指定しておくと役に立ちます。
この指定は、次のような場合の利用が考えられます。ただし、現在は利用できていないものが多く、実際には将来的に考えられる利用方法だともいえます。

- 指定された言語の慣習に従った、より正しい表示をすることが可能になる(例えば、ブラウザが自動的にその言語に合った引用符を選択する場合などに利用できる)
- 別言語バージョンがある場合、Content Negotiationによって自動的に自分の望む言語のページが送信されるようになる(W3Cのページには、このような記述があるが、現在のところ、実際には拡張子でのみ判断可能な模様)
- サーチエンジンのデータを言語別に利用できるようになる
- スピーチシンセサイザなどが正しい発音やアクセントで音声出力できるようになる
- スペルチェッカーや文法チェッカーが適切にチェックすることが可能になる
- 自動翻訳をする場合に適切に翻訳することが可能になる

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML lang="ja">
<HEAD>
   ～
</HEAD>
<BODY>
   ～
</BODY>
</HTML>
```

人にやさしいページを作る

HTML付録

iモード対応のホームページ作成

iモード端末用ページについて

「iモードサービス」を利用すると、携帯電話からインターネットに接続して、Webページを見ることができます。しかし、その場合に見ることのできる内容には、いくつかの制限があります。

実際にiモードから見てもらうためには、iモード端末専用のページを作成する必要があると考えた方がよいでしょう。

具体的な制限事項としては、次のようなものがあります。

- 使用できる要素や属性が限られている
- 表示可能な文字コードはシフトJISのみ
- 表示可能な画像は2階調のGIF形式のみ
- 画像を含む1ページのサイズは基本的に2KBまで
- 表示画面は基本的に全角で横8文字・縦6行
  (機種によって異なる。また、スクロールは可能)
- JavaとJavaScriptは使用不可
  (CGIは利用可)

その反面、iモード端末では、次のようなことが可能になっています。

- 独自の「絵文字」が利用できる
- 半角カナが問題なく利用できる
- リンクを利用して電話をかけることができる
  (URLとして「tel:～」形式が利用可能」)
  【例】 `<A href="tel:03-1234-5678">～</A>`
- ショートカットキーとして「0～9・#・＊」が単独で利用できる
  【例】 `<A href="http://～" accesskey="1">～</A>`

iモード端末で利用できる要素・属性一覧

要素名	属性名
ページの基礎となる内容	
HTML	
HEAD	
BODY	
TITLE	
BASE	href
テキスト	
H1～H6	align
P	align
BLOCKQUOTE	
PLAINTEXT	
DIV	align
スタイルとレイアウト	
BR	clear
PRE	
CENTER	
HR	align・size・width
リンク	
A	name・href・accesskey
リスト	
UL	
OL	
LI	
DL	
DT	
DD	
DIR	
MENU	
画像	
IMG	src・align・width・height・hspace・vspace・alt
入力フォーム	
FORM	action・method
INPUT	type・name・size・maxlength・accesskey・value・checked
TEXTAREA	name・rows・cols
SELECT	name・size
OPTION	selected

※より詳しい内容については、索引「ホームページ作成関連リンク」(P.626)に掲載されている「NTT DoCoMo Net」のホームページをご覧ください。

HTML4.0で定義されている特別な文字

「テキスト」の「特別な文字を表示させる」(P.59)では、日本語環境でも使用できる特別な文字について説明しましたが、HTML4.0ではそれ以外にも多くの文字が定義されています。しかし、実際に表示できる文字は、使用しているOSやブラウザのバージョンなどによって異なります。特に日本語環境で利用している場合には、ほとんどが表示できない場合もあります。ここでは、その中でも多くの文字を表示可能なWindows版のInternet Explorer5.0での表示例を示します。

「特別な文字」は、HTML4.0では文字参照として定義されており、次の3種類に分類されています。

- ISO 8859-1 (欧米：Latin 1)
- ギリシャ文字・シンボル・数学記号
- その他の特別な文字

ここに定義されている文字を表示させる場合、「&キーワード;」という形式と、「&#番号;」という2種類の形式を利用することができます。キーワードについては、大文字と小文字は別の文字として扱われますので注意してください。なお、キーワード形式では表示されないものでも、番号形式で指定すると表示される場合があります。

ISO 8859-1 (Latin 1)

	キーワード	#番号		キーワード	#番号		キーワード	#番号
			¡	¡	¡	¢	¢	¢
£	£	£	¤	¤	¤	¥	¥	¥
¦	¦	¦	§	§	§	¨	¨	¨
©	©	©	ª	ª	ª	«	«	«
¬	¬	¬				®	®	®
¯	¯	¯	°	°	°	±	±	±
²	²	²	³	³	³	´	´	´
µ	µ	µ	¶	¶	¶	·	·	·
¸	¸	¸	¹	¹	¹	º	º	º
»	»	»	¼	¼	¼	½	½	½
¾	¾	¾	¿	¿	¿	À	À	À
Á	Á	Á	Â	Â	Â	Ã	Ã	Ã
Ä	Ä	Ä	Å	Å	Å	Æ	Æ	Æ
Ç	Ç	Ç	È	È	È	É	É	É
Ê	Ê	Ê	Ë	Ë	Ë	Ì	Ì	Ì
Í	Í	Í	Î	Î	Î	Ï	Ï	Ï

Ð	Ð	Ð	Ñ	Ñ	Ñ	Ò	Ò	Ò
Ó	Ó	Ó	Ô	Ô	Ô	Õ	Õ	Õ
Ö	Ö	Ö	×	×	×	Ø	Ø	Ø
Ù	Ù	Ù	Ú	Ú	Ú	Û	Û	Û
Ü	Ü	Ü	Ý	Ý	Ý	Þ	Þ	Þ
ß	ß	ß	à	à	à	á	á	á
â	â	â	ã	ã	ã	ä	ä	ä
å	å	å	æ	æ	æ	ç	ç	ç
è	è	è	é	é	é	ê	ê	ê
ë	ë	ë	ì	ì	ì	í	í	í
î	î	î	ï	ï	ï	ð	ð	ð
ñ	ñ	ñ	ò	ò	ò	ó	ó	ó
ô	ô	ô	õ	õ	õ	ö	ö	ö
÷	÷	÷	ø	ø	ø	ù	ù	ù
ú	ú	ú	û	û	û	ü	ü	ü
ý	ý	ý	þ	þ	þ	ÿ	ÿ	ÿ

ギリシャ文字・シンボル・数学記号

	キーワード	#番号		キーワード	#番号		キーワード	#番号
ƒ	ƒ	ƒ	Α	Α	Α	Β	Β	Β
Γ	Γ	Γ	Δ	Δ	Δ	Ε	Ε	Ε
Ζ	Ζ	Ζ	Η	Η	Η	Θ	Θ	Θ
Ι	Ι	Ι	Κ	Κ	Κ	N	Λ	Λ
Μ	Μ	Μ	Ν	Ν	Ν	Ξ	Ξ	Ξ
Ο	Ο	Ο	Π	Π	Π	Ρ	Ρ	Ρ
Σ	Σ	Σ	Τ	Τ	Τ	Υ	Υ	Υ
Φ	Φ	Φ	Χ	Χ	Χ	Ψ	Ψ	Ψ
Ω	Ω	Ω	α	α	α	β	β	β
γ	γ	γ	δ	δ	δ	ε	ε	ε
ζ	ζ	ζ	η	η	η	θ	θ	θ
ι	ι	ι	κ	κ	κ	σ	λ	λ
μ	μ	μ	ν	ν	ν	ξ	ξ	ξ
ο	ο	ο	π	π	π	ρ	ρ	ρ
ς	ς	ς	σ	σ	σ	τ	τ	τ
υ	υ	υ	φ	φ	φ	χ	χ	χ
ψ	ψ	ψ	ω	ω	amp;969;	ϑ	ϑ	ϑ
ϒ	ϒ	ϒ	ϖ	&piv	ϖ	•	•	•
…	…	…	′	′	′	″	″	″
	‾	‾	⁄	⁄	⁄		℘	℘
	ℑ	ℑ		ℜ	ℜ	™	™	™

	ℵ	ℵ	←	←	←	↑	↑	↑
→	→	→	↓	↓	↓		↔	↔
	↵	↵		⇐	⇐		⇑	⇑
⇒	⇒	⇒		⇓	⇓	⇔	⇔	⇔
∀	∀	∀	∂	∂	∂	∃	∃	∃
	∅	∅	∇	∇	∇	∈	∈	∈
	∉	∉	∋	∋	∋		∏	∏
Σ	∑	∑		−	−		∗	∗
√	√	√	∝	∝	∝	∞	∞	∞
∠	∠	∠	∧	∧	∧	∨	∨	∨
∩	∩	∩	∪	∪	∪	∫	∫	∫
∴	∴	∴		∼	∼		≅	≅
	≈	≈	≠	≠	≠	≡	≡	≡
	≤	≤		≥	≥	⊂	⊂	⊂
⊃	⊃	⊃		⊄	⊄	⊆	⊆	⊆
⊇	⊇	⊇		⊕	⊕		⊗	⊗
⊥	⊥	⊥		⋅	⋅		⌈	⌈
	⌉	⌉		⌊	⌊		⌋	⌋
	⟨	〈	◊	⟩	〉		◊	◊
♠	♠	♠	♣	♣	♣	♥	♥	♥
♦	♦	♦						

その他の特別な文字

	キーワード	#番号		キーワード	#番号		キーワード	#番号
"	"	"	&	&	&	<	<	<
>	>	>	Œ	Œ	Œ	œ	œ	œ
Š	Š	Š	š	š	š	Ÿ	Ÿ	Ÿ
ˆ	ˆ	ˆ	˜	˜	˜			
						‥	‌	‌
	‍	‍		‎	‎		‏	‏
–	–	–	—	—	—	‘	‘	‘
’	’	’	‚	‚	‚	“	“	“
”	”	”	„	„	„	†	†	†
‡	‡	‡	‰	‰	‰	‹	‹	‹
›	›	›	€	€	€			

※このサンプルでは、Internet Explorer5.0を使い、最も一般的と思われる環境(Windows98)・設定(文字コードは「シフト JIS」・フォントは「MS ゴシック」)で、番号形式で文字を指定して表示させました。表示されない、又は文字化けしてしまう文字に関しては、空欄にしてあります。
詳細な使用方法は、「テキスト」の「特別な文字を表示させる」(P.59)を参照してください。

詳解
HTML
&
Java Script
辞典

JavaScript

1. JavaScriptとは?

JavaScriptとは

JavaScriptとは、Netscape社がWebページの処理能力を高めるために開発したLiveScriptを元に、Netscape社とSun社が共同で開発したスクリプト言語で、現在Netscape Navigator 2.0以降のブラウザとInternet Explorer3.0以降のブラウザで対応されています。

JavaScriptを使うことにより、Webページを動的に変化させたり、今までCGIなどで行う必要があった処理の一部を、Webページ上で行うことが可能になります。

仕様にJavaと似た部分があり、JavaScriptが実行できる環境(ブラウザ)さえあれば、OSが違っていても同じように動く(ことを期待できる)プログラムを書ける点などもJavaと似ているといえます。けれども、基本的にJavaとは別物と考えた方がよいでしょう。

Javaとの最大の違いは、コンパイルする必要がないという点です。HTML文章内に直接JavaScriptを記述し、そのファイルをブラウザで読み込むことによって、手軽にスプリクトを実行できる点があげられます。

JavaScriptの種類

JavaScriptは、現在4種類のバージョンが公開されています。JavaScriptのより新しいバージョンは、一部の変更点を除いて、古いバージョンのJavaScriptのすべてに対応しています。

それぞれのバージョンの特徴と対応ブラウザは、次の通りです。

■JavaScript1.0

ブラウザのウィンドウを操作するwindowオブジェクトや、日付を取り扱うDateオブジェクトなど、基本的なオブジェクトが追加されました。

Netscape Navigator2.0、Internet Explorer3.0以降のブラウザでサポートされています。

■JavaScript1.1

画像を取り扱うImageオブジェクトがサポートされました。これによりページ表示後に画像が置き換えられるようになり、マウスポインタの動作によって画像を差し換えたり、定期的に画像を差し換えることによってアニメーションの効果を出したりできるようになりました。

Netscape Navigator3.0、Internet Explorer4.0以降のブラウザでサポートされています。

■JavaScript1.2

ディスプレイサイズなどのディスプレイの情報を取得するscreenオブジェクトや、ウィンドウ上に絶対座標でコンテンツの位置を設定したり、重なりを指定することができるlayerオブジェクトなどが追加されました。これにより、ディスプレイ上のウィンドウの位

置や、ウィンドウ内に表示するコンテンツの位置や重なりを細かく設定することができるようになり、更にそれらを動的に変化させることができるようになりました。

Netscape Navigator4.0、Internet Explorer4.0以降のブラウザでサポートされています。

■JavaScript1.3

文字コードの扱いがUNICODEになったほか、日付を取り扱うDateオブジェクトで年号が4桁で表せるようになったり、ミリセコンド単位の時間を扱えるようになったり、といった細かい部分の追加・変更が行われています。これらの処置は、ECMAScriptと互換を取るためのものです。

Netscape Navigator4.06、Internet Explorer5.0以降のブラウザでサポートされています。

この他のJavaScriptと共に覚えておきたいスクリプト言語に、ECMAScriptとJScriptがあります。これらはJavaScriptをベースにして作られた言語で、JavaScriptと互換を取るように配慮されていますが、その仕様や実装には微妙な違いがあり、残念ながら現状では100%互換があるとは言い切れません。

それぞれの言語の特徴は、次の通りです。

■ECMAScript

ヨーロッパの標準化機関であるECMA(EuropeanComputerManufacturersAssociation)が、JavaScript1.1をベースに規定した、インターネットで使用するスクリプト言語の仕様。ECMA-262として仕様が公開されています。

ECMAScriptはあくまでも言語仕様を規定したものであり、その仕様に合わせてどのように実装されるかは、その言語によって変わってきます。

現在、JavaScript1.3とMicrosoft社のJScript3.1が、ECMAScriptと互換が取られていることになっていますが、2つのスクリプト言語にはかなり違う部分があります。

■JScript

Microsoft社が独自に開発した、JavaScript互換のスクリプト言語。Internet Explorerに搭載されているのは、正確にいうとJavaScriptではなく、このJScriptということになります。

Internet Explorer3.0(Macintosh版はInternet Explorer3.1)から実装されており、Internet Explorer3.xにはJavaScript1.0レベルのスクリプトが実行できるJScriptが、Internet Explorer4.xにはJavaScript1.1レベルのスクリプトとJavaScript1.2の一部が実行できるレベルのJScriptが実装されています。

現在仕様が公開されているJScriptはJScript5.0(日本語ページではJScript4)で、これはJavaScript互換というより、ECMAScript互換スクリプトといった方がよく、ECMAScriptに準拠しながらInternet Explorer独自の拡張が施されています。Internet Explorer5.xで採用されています。

ブラウザから見たJavaScriptへの対応状況

Netscape Navigator、Internet Explorer両ブラウザの、各バージョンで対応しているJavaScriptと、JavaScriptを使う上での注意点は、次の通りです。

■Netscape Navigator2.x

JavaScript1.0に対応。

JavaScriptに始めて対応したブラウザ。Macintosh版では、新しいウィンドウを開いた時、表示するページのURLが取得できずにページが表示できない、ループしてJavaScriptを実行するとメモリーエラーが発生する、などの問題がありました。

■Netscape Navigator3.x

JavaScript1.1に対応。

Netscape Navigator2.xのMacintosh版にあった問題点が解消されています。

■Netscape Navigator4.x

JavaScript1.2に対応。

レイヤーなど、Netscape Navigator独自仕様のものもJavaScriptで制御できるようになりましたが、その結果、スクリプトの独自色が強まってしまいました。

また、W3Cによる標準仕様が存在するスタイルシートでもレイヤーと同じようにJavaScriptで制御できますが、Netscape Navigator自身のスタイルシートの実装には、不完全な部分があります。

■Netscape Navigator4.06以降

JavaScript1.3に対応。

文字がユニコードで扱われるようになったため、ページ上やフォーム、ステータス行などにJavaScriptを使用して文字を書き出す場合、表示される文字が文字化けを起こす可能性があります(この問題は、Netscape Navigator4.5日本語版では解消されています)。また、ユニコードは文字が2バイト文字なので、文字数を数えているようなスクリプトは、今までとは違う結果になる場合があります。

■Internet Explorer3.x

JavaScript1.0レベルのスクリプトに対応。

Internet Explorer3.xは、例え同じバージョン番号がついたものであっても細かい改良が加えられており、実行できるJavaScriptも増えてきています。その結果、同じバージョンのInternet Explorerで同じスクリプトを実行したとしても、一方のブラウザは正常に実行でき、一方のブラウザはエラーになる、という現象が発生しています。

Internet Explorer3.xは、JavaScript以外でも、HTMLの表示やセキュリティーの問題など、多くの部分が改良されているので、なるべく最新のものを使用することをお勧めします。

■Internet Explorer4.x

JavaScript1.1レベルのスクリプトに対応。JavaScript1.2レベルのスクリプトも、レイヤーなどNetscape Navigator独自の部分を除き、ほぼ対応。ECMAScript(ECMA-262)の仕様に準拠しているので、正式にはJavaScript1.3には対応していませんが、JavaScript1.3の多くのスクリプトが実行可能です。

その他、文字をユニコードで取り扱うようになったことを始め、ECMAScriptの仕様に準拠しながら、独自の拡張が行われています。スタイルシート関連のタグを始め、Internet Explorerでサポートされているほとんどのタグを、オブジェクトとして取り扱うことが可能です。

■Internet Explorer5.x

JavaScriptのサポートに関しては、Internet Explorer4.xとほぼ同等。

JavaScript1.3にも対応しています。

Internet Explorerが採用しているJScriptは、ブラウザのバージョンが同じものでも、バグの修正・改良・バージョンアップなどが行われる場合があります。

注意

「LANGUAGE」オプションの記述について

現在は、JavaScriptを記述する時に「LANGUAGE」オプションを省略して次のように記述しても、Netscape Navigator、Internet Explorerどちらのブラウザも、<SCRIPT>タグ内に記述されたスクリプトをJavaScriptとして解釈し、実行します。

```
<SCRIPT>
<!---
JavaScriptのソース
//-->
</SCRIPT>
```

しかしながら、JavaScript以外にも、VBScriptなど<SCRIPT>タグ内で使用できるスクリプトがあること。きちんとオプション指定をし、JavaScriptのバージョンを記述することにより、そのバージョンに対応していないブラウザで発生するエラーが簡単に回避できること。更に今後、「LANGUAGE」オプションを指定しなければJavaScriptとして解釈しないブラウザが出て来る可能性があることなどの理由から、「LANGUAGE」オプションは指定すべきでしょう。

2. JavaScriptの記述方法

JavaScriptは、HTMLファイル内にHTMLタグを使ってJavaScriptであることを指定し、そのタグ内にソースコードを記述することによって設定します。

JavaScript関連で使用するタグと、それらのタグを使ってHTMLファイル内へJavaScriptを記述する方法は、次の通りです。

<SCRIPT>タグの使い方

HTMLファイル内にJavaScrtptを記述するには、<SCRIPT>タグを使用します。

<SCRIPT>タグ内の「LANGUAGE」オプション(HTMLパートでは、language属性といっています)指定で、「LANGUAGE="JavaScript"」と記述しておくと、ブラウザはその中に記述されている文字列がJavaScriptであると判断し、実行します。

Imageオブジェクトなど、JavaScript1.1を使ってスクリプトを記述する場合には、「LANGUAGE」オプションの指定を「LANGUAGE="JavaScript1.1"」とします。こうやって指定された<SCRIPT>タグ内のスクリプトは、Netscape Navigator3.0などのJavaScript1.1に対応したブラウザでのみ実行され、Netscape Navigator2.0などのJavaScript1.1未対応のブラウザでは実行されません。

また、layerオブジェクトなど、JavaScript1.2を使用してスクリプトを記述する場合は、「LANGUAGE」オプションの指定を「LANGUAGE="JavaScript1.2」とします。こうやって指定された<SCRIPT>タグ内のスクリプトは、Netscape Navigator4.0などのJavaScript1.1に対応したブラウザでのみ実行されるようになります。

同様に、JavaScript1.3を使ったスクリプトを記述する場合の「LANGUAGE」オプションの指定は、「LANGUAGE="JavaScript1.3」となります。こうやって指定された<SCRIPT>タグ内のスクリプトは、Netscape Navigator4.5などのJavaScript1.3に対応したブラウザでのみ実行されます。

コラム

Netscape NavigatorとInternet ExplorerとECMAScriptの関係

Netscape NavigatorでのECMAScriptのサポートは、JavaScript1.3から、つまりNetscape Navigator4.06からです。それに対してInternet Explorerは、Internet Explorer4.xからECMAScriptをサポートしています。

また、Netscape Navigator4.06以前のNetscape Navigator4.xブラウザでも、一部ECMAScriptをサポートしています。

つまり、Internet Explorer4.xやNetscape Navigator4.06以前のNetscape Navigator4.xは、<SCRIPT>タグ内の「LANGUAGE」オプションの設定で「<SCRIPT LANGUAGE="JavaScript1.3">」と設定されたスクリプトは実行できません。けれども、「getFullYear()」などの、JavaScript1.3に含まれた多くのECMAScriptを実行することができます。

このように、あらかじめ「LANGUAGE」オプションでJavaScriptのバージョンを記述しておけば、記述したバージョンに対応していないブラウザではそのスクリプトは実行されません。これにより、JavaScriptに未対応のブラウザで見た時に起こるエラーを回避することができます。

ブラウザは、基本的に自分の知らないタグを無視します。そのためJavaScriptに未対応のブラウザでJavaScriptを記述したページを見ると、ソース部分が丸見えになってしまいます。

これを防ぐため、スクリプトの1行目を「<!--」に、最終行を「//-->」とすることによって、ソースをコメントアウトします。JavaScript未対応のブラウザで閲覧している人に迷惑をかけないように、「おまじない」とでも考えて、必ず記述するようにしてください。

具体的な<SCRIPT>タグの書き方は、次のようになります。

この「JavaScript1.xのソース」の部分に、実際のスクリプトを記述します。スクリプトは、HTMLファイルに記述されている上から順に評価され、より下に記述されているものが優先して実行されます。

・JavaScriptの記述法

```
<SCRIPT LANGUAGE="JavaScript">
<!--
JavaScript1.0のソース
//-->
</SCRIPT>
```

・JavaScript1.1の記述法

```
<SCRIPT LANGUAGE="JavaScript1.1">
<!--
JavaScript1.1のソース
//-->
</SCRIPT>
```

・JavaScript1.2の記述法

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
JavaScript1.2のソース
//-->
</SCRIPT>
```

・JavaScript1.3の記述法

```
<SCRIPT LANGUAGE="JavaScript1.3">
<!--
JavaScript1.3のソース
//-->
</SCRIPT>
```

JavaScript

JavaScript について

<NOSCRIPT>タグの使い方

<NOSCRIPT>タグは、JavaScriptを無効にしているか、対応していないブラウザを使っているユーザーに対してメッセージを表示する時に使用します。

<NOSCRIPT>タグは、<SCRIPT>～</SCRIPT>タグ外で使用し、JavaScriptが有効の時には<NOSCRIPT>～</NOSCRIPT>タグ内の記述は表示されません。

具体的な<NOSCRIPT>タグの書き方は、次のようになります。

```
<NOSCRIPT>
このページではJavaScriptが使われています....。
</NOSCRIPT>
```

コメントの書き方

JavaScript内でのコメントを書くには、次の2通りの方法があります。

「//」はそれ以降の1行が、「/*～*/」では間に挟まれた文字列が、コメントとして扱われます。

ソースコードをコメントアウトする時に最後の行に使用する、「//-->」の「//」の部分もこれに当たります。

具体的なコメントの書き方は、次のようになります。

```
<SCRIPT LANGUAGE="JavaScript">
<!--
//この1行はコメントとなります。
/*これに囲まれている部分
  はコメントとなります。*/
//-->
</SCRIPT>
```

文の区切り方

「;」は文の区切りを表します。JavaScriptでは、たとえ改行があったとしても、「;」によって文を区切っていない限り、1つの文として扱われます。

基本的に、「;」を省略してもJavaScriptは自動的に文の区切りを判断し、スクリプトは正常に処理されますが、ソースを分かりやすくする意味も含めて、文は「;」で区切っておくことをお勧めします。

JavaScriptの外部呼び込み方法

JavaScriptは、HTMLファイル内に直接記述する方法以外に、外部にスクリプトを記述したファイルを置き、<SCRIPT>タグ内で「SRC」オプションを使用してURLを指定することによって、それを読み込んで実行することができます。指定したファイル内には、前後の<SCRIPT>タグを省略したJavaScriptのソースコードを記述します。この時、ファイル名の拡張子は「.js」とします。

```
<SCRIPT SRC="URL"></SCRIPT>
```

この方法は、スクリプトのソースコードを見られにくくしたり(完全に隠蔽することはできません)、複数のページで同一のスクリプトを使用する場合に特に有効です。例えば、ファイルの更新日時を複数のWebページ上に表示するような場合、必要な全ページに1つ1つスクリプトを記述するのではなく、次のようにファイルの最終更新日時を書き出すスクリプトを書いて、適当な名前を付けた拡張子「.js」のファイルを1つ作ります。後は、すべてのページに、そのファイルのURLを指定した<SCRIPT>タグを、表示したい位置に記述するだけで済むのです。

```
document.write("Last update:",document.lastModified)
```

この方法は、Netscape Navigator3.0以降、Internet Explorer4.0以降から使用可能です。

本来ならば、<SCRIPT>タグ同様に「LANGUAGE」オプションを使用して、JavaScriptのバージョンを記述することにより、そのJavaScriptのバージョンに対応がとられた「.js」ファイルしか読み込まないようにできるはずなのですが、現在その機能は使用できません。

また、「.js」ファイルを置くWebサーバーには、MIMEタイプの設定をしておく必要があります。

注意

MIMEタイプの設定について

JavaScriptの外部読み込みをするためには、あらかじめサーバーにMIMEタイプの設定がされていなければなりません。

例えば、WebサーバーにNCSAhttpdやApacheが使われていて、ユーザーのホームディレクトリで「.htaccess」ファイルにより設定が許されているような場合は、その中に次のように記述することによって設定します。

```
AddType application/x-javascript .js
```

MIMEタイプの設定には、このほかにもいろいろな方法があり、プロバイダによっては「.htaccess」ファイルによる設定を行わなくても、MIMEタイプがされている場合もあります。

MIMEタイプの設定に関しては、Webサーバーの運用に関わってくる問題も含まれるので、サーバー管理者の方と相談するようにしてください。

3. オブジェクト・プロパティ・メソッド

オブジェクト

JavaScriptでは、ブラウザの各部品や情報をオブジェクトとして取り扱うことができます。そして、このオブジェクトの値を変更したり、値を調べてその結果によって違った処理を設定することで、ブラウザを動的に変更することができるのです。

JavaScriptのオブジェクトを大きく分けると、ブラウザ自身が本来持っている部品や情報を取り扱うナビゲータオブジェクトと、独自に組み込まれたビルトイン(組み込み)オブジェクトの2種類になります。

・ナビゲータオブジェクト

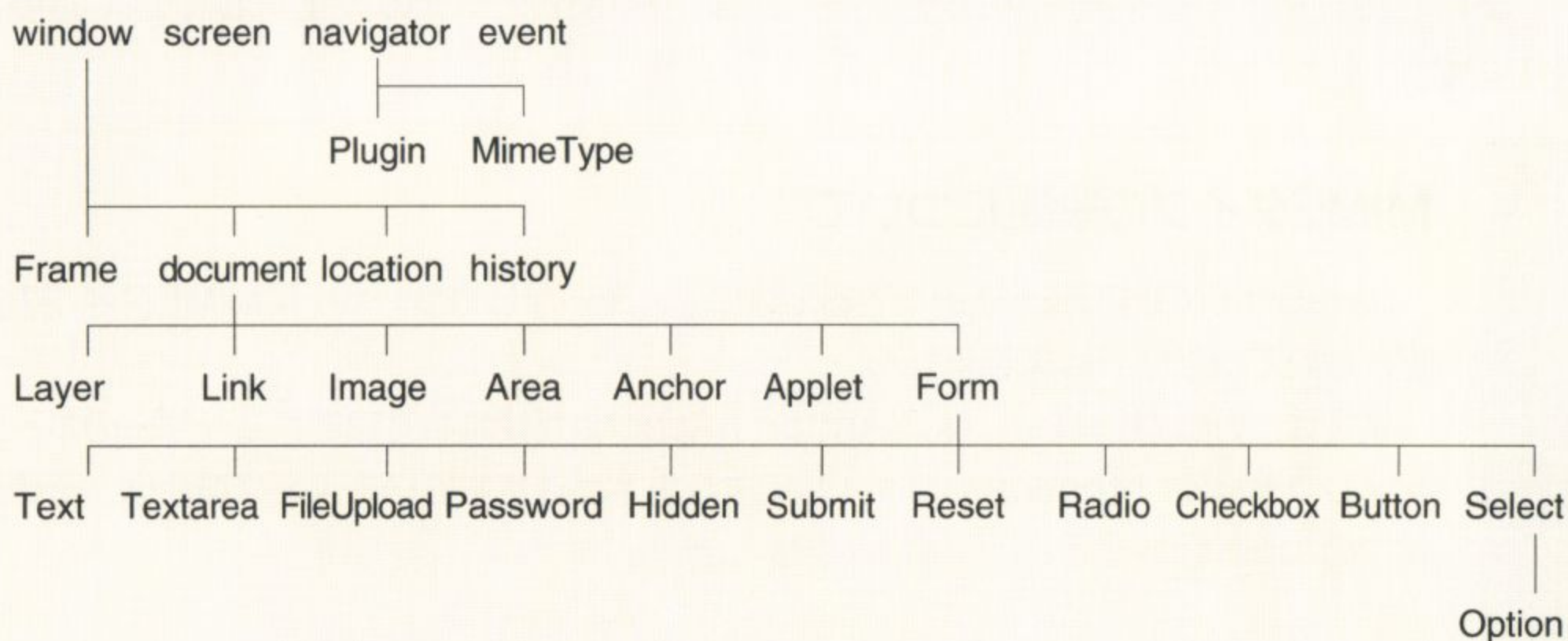
ブラウザ自体の名前やバージョンといった情報や、ブラウザに表示されるドキュメント・画像・フォームなど、ブラウザがあらかじめ持っている部品を取り扱うオブジェクトのことを、ナビゲータオブジェクトといいます。

ナビゲータオブジェクトには階層関係があり、使用する時はその階層関の上から順番に「.」で区切って記述します。

例えばdocumentオブジェクトは、windowオブジェクトの1つ下の階層にあるオブジェクトなので、「window.document」として表します。ただし、1番上の階層になるwindowオブジェクトは、省略することができます。

ナビゲータオブジェクトの階層関係は、下図の通りです。

図「ナビゲータオブジェクトの階層」

- window
  - Frame
  - document
    - Layer
    - Link
    - Image
    - Area
    - Anchor
    - Applet
    - Form
      - Text
      - Textarea
      - FileUpload
      - Password
      - Hidden
      - Submit
      - Reset
      - Radio
      - Checkbox
      - Button
      - Select
        - Option
  - location
  - history
- screen
- navigator
  - Plugin
  - MimeType
- event

・ビルトインオブジェクト

ブラウザ自信が本来持っているオブジェクトの他に、JavaScriptがブラウザに独自に組み込んでいるオブジェクトを、ビルトインオブジェクトといいます。

JavaScriptでは、日付・時間などの時間を取り扱うオブジェクトや、文字列の操作を行うオブジェクトなど、多くのビルトインオブジェクトが用意されています。

ユーザー独自のオブジェクトを作成したり、ビルトインオブジェクトを使用する時は、new演算子を使います。new演算子によるオブジェクトの作成(インスタンスの作成)は、次の書式で行います。そしてそれ以降は、「オブジェクト名」で指定した名前を使って、オブジェクトの操作を行うことができるようになります。

```
オブジェクト名 = new オブジェクトの型(値)
```

このオブジェクト名は、一定の条件下において、ユーザー側で自由に設定することができます。

プロパティ

オブジェクトは多くの属性を持っており、この属性のことをプロパティといいます。

プロパティもまた、オブジェクトです。例えばdocumentオブジェクトは、それ自体がオブジェクトであるのと同時に、windowオブジェクトのプロパティであるともいえます。

JavaScriptでは、プロパティは、オブジェクトの後に「.」で区切って設定します。

```
オブジェクト.プロパティ
```

また、プロパティの中には、ユーザーが値を設定することができるものがあります。そのようなプロパティに値を設定する場合は、次の書式で設定します。

```
オブジェクト.プロパティ=値
```

各オブジェクトが、どのようなプロパティを持っているかは、リファレンスの「ナビゲータオブジェクト」(P.553〜)・「ビルトインオブジェクト」(P.568〜)を参照してください。

メソッド

メソッドは、オブジェクトに対して動作を指定します 。

メソッドは、次のようにオブジェクトの後に「.」で区切って設定します。また、オブジェクトに値を設定する時には、「()」内に値を設定することによって行います。

```
オブジェクト.メソッド(値)
```

JavaScriptの各オブジェクトがどのようなメソッドを持っているかは、リファレンスの「ナビゲータオブジェクト」(P.553〜)・「ビルトインオブジェクト」(P.568〜)を参照してください。

4. イベントハンドラ

ユーザーやスクリプトによってページがロードされたり、オブジェクトがクリックされたりというような、特定の動作が起こったタイミングをイベントといいます。JavaScriptでは、イベントの発生を取得して、そのタイミングでスクリプトの実行を開始することができます。

このイベントの取得を行うものを、イベントハンドラといいます。

イベントハンドラの設定は、そのイベントハンドラが設定可能なオブジェクトのHTMLタグ内に、次のような書式で設定します。

```
イベントハンドラ名=スクリプト、又は関数
```

JavaScriptで用意されているイベントハンドラと、そのイベントハンドラがどのようなイベントを取得し、どのオブジェクトに対応しているかは、リファレンスの「イベントハンドラ」(P.547～)を参照してください。

5. イベントタイプ

JavaScript1.2からは、イベントをオブジェクトとして捕らえるeventオブジェクトが追加されました。これにより、イベントを取得したいオブジェクトに対して取得するイベントのタイプを設定することで、そのオブジェクト上のどこででもイベントの発生を取得することができるようになりました。

イベントの取得は次の用法で設定します。

```
オブジェクト.イベントタイプ=関数名、又はスクリプト
```

JavaScriptで設定できるイベントタイプと、そのイベントタイプが設定できるオブジェクト、イベントタイプで取得できる値、つまりeventオブジェクトのプロパティにどのようなものがあるかは、リファレンスの「イベントタイプ」(P.550)を参照してください。

6. JavaScriptで取り扱える型の種類

プロパティやメソッドに設定する値はもちろん、スクリプトが返す値や変数・定数など、JavaScriptで取り扱う値(データ)は、必ずなんらかのデータ型を持っています。

JavaScriptで取り扱える型にどのようなものがあるかは、リファレンスの「JavaScriptで取り扱える型の種類」(P.538)を参照してください。

7. 関数

一連の処理手続をまとめて名前を付けたものを、関数といいます。

JavaScriptにおける関数は、大きく分けて関数の処理を定義する部分と、関数を呼び出す部分の2つの部分からできています。

関数の処理の定義は、次のように記述します。

```
function 関数名(引数,引数,...){ 処理 }
```

この関数名は、一定の条件下でユーザー側が自由に設定することができます。

関数の呼び出しはページ上やイベントハンドラ内で行い、関数の処理を呼び出したい部分に「関数名(引数,引数,...)」と記述して設定します。こうしておけば、ページが読み込まれた時や特定のイベントが発生した時に関数の処理が呼び出だされ、定義した関数の処理が実行されます。

引数とは、その名の通り、関数の処理に引き渡す値のことをいいます。関数の処理に値を引き渡す必要がない時は、引数を設定する必要はありません。

通常、関数の処理の定義は<HEAD>タグ内で行い、関数の処理の呼び出しは<BODY>タグ部で行います。これは、関数の定義が終わらないうちに関数が呼び出されることを防ぐためです。

8. ビルトイン関数

JavaScriptには始めから定義されている関数があり、これをビルトイン関数といいます。

ビルトイン関数はオブジェクトに依存することなく、スクリプト内のどこからでも使用することができます。

JavaScriptで用意されているビルトイン関数にどのようなものがあるかは、リファレンスの「ビルトイン関数(top-level関数)」(P.574)を参照してください。

9. 変数・定数

変数とは、値を入れておく箱のようなものと考えればいいでしょう。文字列や数値、オブジェクト・プロパティ・メソッド、あるいは条件式などで設定する式や変数などを、変数として設定することができます。

変数は、次のようにして設定します。

```
var 変数名 = 値
```

変数名は、一定の条件下でユーザー側が自由に設定することができます。また、「var」は省略可能です。

このようにして設定すると、右辺の「変数名」に「値」が代入され、以降は「変数名」を使用することによって値を取り出すことができます。

更に、後から同じ「変数名」に値を設定し直すことによって、変数の値を自由に変化させることもできます。

このように、値を設定し直すことによって絶えず値が変化する変数に対し、変数に設定する値の方は変化することはありません。このことから、変数に代入する値のことを定数(リテラル)といいます。

10. オブジェクト、関数、変数などに設定可能な名前

オブジェクト名・関数名・引数名・変数名・定数名は、以下の条件において、ユーザー側で自由に名前を定義することができます。

1. 大文字・小文字のアルファベット、あるいはアンダースコア"_"で始まる文字列
   (hamba,HAMBA, _hambaなど)
2. 日本語(2バイト文字)は使用できない
3. スペース・コンマ・疑問符・引用符は使用できない
4. 文字列内に数値を入れることは可能だが、数値を先頭にすることはできない
   ("Hamba1"や、"f_3hamba"は可能だが"6hamba"は不可)
5. 大文字・小文字は区別される
   ("HAMBA"と"hamba"は区別される)
6. 予約語は使用できないが、以上の条件を満たし、予約語を含む文字列
   ("default"は不可だが、"Setdefault"は可)

予約語

システム側であらかじめ予約されている文字を予約語といい、オブジェクト名・関数名・引数名・変数名・定数名として使用することはできません。

予約語には、次のようなものがあります。

abstract	boolean	break	byte	case
catch	char	class	const	continue
default	delete	do	double	else
extends	false	final	finally	float
for	function	goto	if	implements
import	in	instanceof	int	interface
long	native	new	null	package
private	protected	public	return	short
static	super	switch	synchronized	this
throw	throws	transient	true	try
typeof	var	void	while	with

11. 演算子

値の計算や、比較などに用いる記号のことを演算子といいます。

JavaScriptで使用できる演算子にどのようなものがあるかは、リファレンスの「演算子」(P.539～)を参照してください。

12. JavaScriptの命令文(ステートメント)

JavaScriptでは、if文やfor文を始め、多くの命令文を使うことができます。

JavaScriptで使うことができる命令文と、その使い方については、リファレンスの「JavaScriptの命令文(ステートメント)」(P.542～)を参照してください。

JavaScript

ブラウザ名を取得する

navigator.appName [プロパティ]

Internet Explorer	Netscape Navigator
*ブラウザ名を取得する ブラウザ名:Microsoft Internet Explorer	*ブラウザ名を取得する ブラウザ名:Netscape

解説

「appName」プロパティは、ブラウザ名を取得します。Netscape Navigatorは「Netscape」を、Internet Explorerは「Microsoft Internet Explorer」を返します。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("ブラウザ名:",navigator.appName)
//--->
</SCRIPT>
```

JavaScript

ブラウザのコード名を取得する

navigator.appCodeName [プロパティ]

Internet Explorer	Netscape Navigator
*ブラウザのコード名を取得する コード名:Mozilla	*ブラウザのコード名を取得する コード名:Mozilla

解説

「appCodeName」プロパティは、ブラウザのコード名を取得します。Netscape Navigatorは「Mozilla」を、Internet Explorerは「Mozilla/3.0 (compatible; MSIE 3.01;Mac_PowerPC)」や「MSIE」、「Mozilla」などを返します。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("コード名:",navigator.appCodeName)
//--->
</SCRIPT>
```

JavaScript

ブラウザのバージョンを取得する

navigator.appVersion 【プロパティ】

NN4.0 NN4.06 IE4.0 IE5.0

Internet Explorer	Netscape Navigator
*ブラウザのバージョンを取得する バージョン:4.0 (compatible; MSIE 5.0; Windows 98)	*ブラウザのバージョンを取得する バージョン:4.6 [ja] (Win98; I)

解説 「appVersion」プロパティは、ブラウザのバージョンを取得します。
バージョン番号の後ろに、OS名、インターナショナル版・U.S版などの種別、CPUの種類、などの情報が付加されます。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("バージョン:",navigator.appVersion)
//--->
</SCRIPT>
```

JavaScript

ブラウザのユーザーエージェントを取得する

navigator.userAgent 【プロパティ】

Internet Explorer	Netscape Navigator
*ブラウザのユーザーエージェントを取得する ユーザーエージェント:Mozilla/4.0 (compatible; MSIE 5.0; Windows 98)	*ブラウザのユーザーエージェントを取得する ユーザーエージェント:Mozilla/4.6 [ja] (Win98; I)

解説 「userAgent」プロパティは、ブラウザのユーザーエージェントを取得します。
ユーザーエージェントとは、HTTPのヘッダー部分に付けられている文字列です。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("ユーザーエージェント:",navigator.userAgent)
//--->
</SCRIPT>
```

JavaScript

ユーザーのプラットホームのタイプを取得する

navigator.platform [プロパティ]

*ユーザーのプラットホームのタイプを取得する

お使いのマシンのタイプ:Win32

解説

「platform」プロパティは、ユーザーのプラットフォームのタイプを持っています。サンプルでは、ユーザーの環境に合わせて「Win32」や「MacPPC」といった値を表示します。
JavaScript1.2で追加された、読み出し専用のプロパティです。

Sample

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("お使いのマシンのタイプ:",navigator.platform)
//-->
</SCRIPT>
```

JavaScript

ブラウザの使用言語を取得する

navigator.language [プロパティ]

*ブラウザの使用言語を取得する

使用言語:ja

解説

「language」プロパティは、ブラウザの「ja」や「en」などのLANG属性を返します。
「編集」メニューの「設定....」-「Navigator」-「言語」から設定することができます。
JavaScript1.2で追加された、読み出し専用のプロパティです。
このプロパティはInternet Explorerでは未対応です。

Sample

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("使用言語:",navigator.language)
//-->
</SCRIPT>
```

ブラウザの判別をする

navigator.appName [プロパティ]
navigator.appVersion [プロパティ]

Internet Explorer

*ブラウザの判別をする

お使いのブラウザはExplorer5.xですね

Netscape Navigator

*ブラウザの判別をする

お使いのブラウザはNetscape4.xですね

解説

サンプルは、ブラウザ名とバージョンの1番始めの文字を検索し、それを元にブラウザを判別してブラウザ名を書き出しています。
Internet Explorer3.xは、バージョンとして「2.0」を返すものがあるので、その点も考慮しています。また、Internet Explorer5.0は、現在バージョンとして「4.0」を返すので、バージョン情報の中から「MSIE 5」という文字列を検索することによって判断しています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<SCRIPT LANGUAGE="JavaScript">
<!---
if( navigator.appName.charAt(0)=="N" ){
    if(navigator.appVersion.charAt(0)==2){ document.write
("お使いのブラウザはNetscape2.xですね") }
    if(navigator.appVersion.charAt(0)==3){ document.write
("お使いのブラウザはNetscape3.xですね") }
    if(navigator.appVersion.charAt(0)==4){ document.write
("お使いのブラウザはNetscape4.xですね") }
}
```

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript

```
if( navigator.appName.charAt(0)=="M" ){
    if(navigator.appVersion.charAt(0)==2){ document.write
("お使いのブラウザはExplorer3.xですね") }
    if(navigator.appVersion.charAt(0)==3){ document.write
("お使いのブラウザはExplorer3.xですね") }
    if(navigator.appVersion.charAt(0)==4){
           if (navigator.appVersion.indexOf("MSIE 5") !=
-1) {document.write("お使いのブラウザはExplorer5.xですね")}
           else { document.write("お使いのブラウザはExplorer4.xで
すね") }
      }
    if(navigator.appVersion.charAt(0)==5){ document.write
("お使いのブラウザはExplorer5.xですね") }
}
//-->
</SCRIPT>
</BODY>
</HTML>
```

charAt()→「stringオブジェクト」の「n番目の文字を抜き出す」:P.487参照
indexOf()→「stringオブジェクト」の「先頭から文字列を検索する」:P.490参照

JavaScript

NN3.0
NN4.0
NN4.06

Javaが使えるかどうか判断する

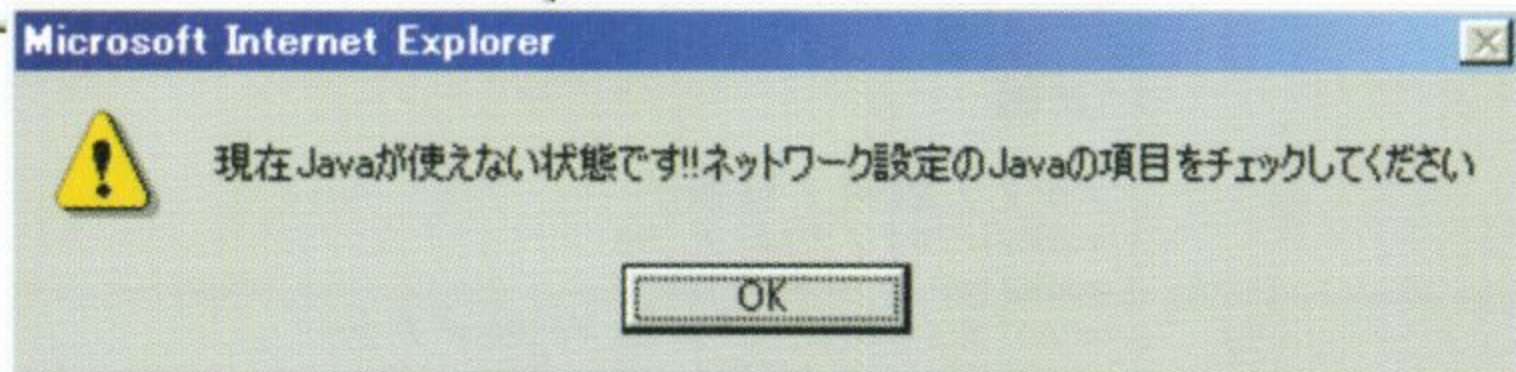
navigator.javaEnabled() [メソッド]

*Javaが使えるかどうか判断する

Javaが使えます

Microsoft Internet Explorer

現在Javaが使えない状態です!!ネットワーク設定のJavaの項目をチェックしてください

OK

解説

サンプルは、「javaEnabled()」メソッドでJavaが使える状態になっているかを判断し、使えれば「Javaが使えます」と書き出し、使えない状態であれば警告用のダイアログボックスを開きます。
JavaScript1.1で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*Javaが使えるかどうか判断する<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
if (navigator.javaEnabled()) { document.write("Java が使え
ます") }
    else window.alert("現在Javaが使えない状態です!!ネットワーク設定の
Javaの項目をチェックしてください")
//--->
</SCRIPT>
</BODY>
</HTML>
```

alert()→「windowオブジェクト」の「警告用のダイアログボックスを開く」:P.287参照

NN3.0
NN4.0
NN4.06

JavaScript

使用可能なMIMEのタイプを取得する

navigator.mimeTypes	[オブジェクト(配列)]
navigator.mimeTypes[n].type	[プロパティ]
navigator.mimeTypes[n].description	[プロパティ]
navigator.mimeTypes[n].suffixes	[プロパティ]

*使用可能なMIMEのタイプを取得する

186**個**

タイプ / 説明 / 拡張子
audio/x-rmf / RMF / rmf
audio/rmf / RMF / rmf
video/msvideo / Video for Windows / avi
audio/x-midi / MIDI / mid, midi
audio/nspaudio / Netscape Packetized Audio / la, lma

解説

サンプルでは、「length」プロパティでMIMEタイプの数を出し、そのブラウザで利用可能なMIMEタイプの一覧を作成しています。
「type」プロパティはMIMEのタイプを、「description」プロパティはその詳細を、「suffixes」プロパティは拡張子をそれぞれ返します。mimeTypesオブジェクトには、その他に「enablePlugin」(プラグインを使うための名前を返す)というプロパティがあります。JavaScript1.1で追加されたプロパティです。

Sample

```
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
var L = navigator.mimeTypes.length;
document.write( L );
document.write("個".bold());
document.write("<P>");
document.write("タイプ / 説明 / 拡張子".bold());
document.write("<BR>");
for(i=0; i<L; i++){
document.write(navigator.mimeTypes[i].type);
document.write(" / ".bold());
document.write(navigator.mimeTypes[i].description);
document.write(" / ".bold());
document.write(navigator.mimeTypes[i].suffixes);
document.write("<BR>");
}
//--->
</SCRIPT>
```

bold()→「stringオブジェクト」の「太字(ボールド)にする」:P.479参照
length→「複数のオブジェクトで使用できるプロパティ・メソッド」の「オブジェクト(配列)の数を取得する」:P.512参照

使用可能なプラグインを取得する

NN3.0 NN4.0 NN4.06

navigator.plugins ［オブジェクト(配列)］
navigator.plugins[*n*].name ［プロパティ］
navigator.plugins[*n*].filename ［プロパティ］
navigator.plugins[*n*].description ［プロパティ］

*使用可能なプラグインを取得する

6個

名前 / ファイル名 / 説明
Headspace Beatnik Player Stub V1.0.0.1 / C:¥PROGRAM FILES¥NETSCAPE¥COMMUNICATOR¥PROGRAM¥plugins¥NPBeatSP.d
Player Stub for Netscape Communicator
Shockwave Flash / C:¥PROGRAM FILES¥NETSCAPE¥COMMUNICATOR¥PROGRAM¥plugins¥npswf32.dll /

解説

サンプルでは、「length」プロパティでプラグインの数を出し、そのブラウザで使用可能なプラグインの一覧を作成しています。
「name」プロパティはプラグインの名前を、「filename」プロパティはファイル名を、「description」プロパティはその詳細をそれぞれ返します。
JavaScript1.1で追加されたプロパティです。

Sample

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*使用可能なプラグインを取得する<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
var L = navigator.plugins.length
document.write( L );
document.write("個".bold());
document.write("<P>");
document.write("名前 / ファイル名 / 説明".bold());
document.write("<BR>");
for(i=0; i<L; i++){
document.write(navigator.plugins[i].name);
document.write(" / ".bold());
document.write(navigator.plugins[i].filename);
document.write(" / ".bold());
document.write(navigator.plugins[i].description);
document.write("<BR>");
}
//--->
</SCRIPT>
</BODY></HTML>
```

bold()→「stringオブジェクト」の「太字(ボールド)にする」:P.479参照
length→「複数のオブジェクトで使用できるプロパティ・メソッド」の「オブジェクト(配列)の数を取得する」:P.512参照

NN3.0
NN4.0
NN4.06

JavaScript

プラグインがインストールされているかチェックする

navigator.plugins [オブジェクト(配列)]
navigator.plugins[*n*].name [プロパティ]

*プラグインがインストールされているかチェックする

Default Plug-inがインストールされています

解説

サンプルでは、プラグイン配列の中に「Default Plug-in」の一部の「efault」という文字列が含まれているかどうかを検索して、含まれていれば文字を書き出し、含まれていなければ警告用のダイアログボックスを開いています。
JavaScript1.1で追加されたプロパティです。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
function CPlug(CP) {
    for(i=0; i<navigator.plugins.length; i++) {
        if (navigator.plugins[i].name.indexOf(CP) != -1)
return true
     }
     return false
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*プラグインがインストールされているかチェックする<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
if (CPlug("efault")) { document.write("Default Plug-inが
インストールされています") }
    else window.alert("Default Plug-inがインストールされていませ
ん!!")
//--->
</SCRIPT>
</BODY></HTML>
```

alert()→「windowオブジェクト」の「警告用のダイアログボックスを開く」:P.287参照
indexOf()→「stringオブジェクト」の「先頭から文字列を検索する」:P.490参照
length→「複数のオブジェクトで使用できるプロパティ・メソッド」の「オブジェクト(配列)の数を取得する」:P.512参照

その他のnavigatorオブジェクト

refresh () [メソッド]

解説

plugins配列をリフレシュします。この時にtrue(真)を返すと、それ以降<EMBED>タグで指定してプラグインを使用することができ、false(偽)を返した時は、プラグインの配列の作り直しのみを行います。
JavaScript1.1で追加されたpluginsオブジェクトのメソッドです。

用法

```
navigator.plugins.refresh([true|false])
```

preference() [メソッド]

解説

クライアントのpreferenceの設定を行います。
Signed Script内で使用可能です。
JavaScript1.2で追加されたメソッドです。

用法

```
navigator.preference(設定名)
navigator.preference(設定名, 設定値)
```

設定項目	設定名	設定値
Automatically load images	general.always_load_images	true I false
Enable Java	security.enable_java	true I false
Enable JavaScript	avascript.enabled	true I false
Enable style sheets	browser.enable_style_sheets	true I false
Enable autoinstall	autoupdate.enabled	true I false
Accept all cookies	network.cookie.cookieBehavior	0
Accept only cookies that getsent back to the originating server	network.cookie.cookieBehavior	1
Disable cookies	network.cookie.cookieBehavior	2
Warn before accepting cookie	network.cookie.warnAboutCookies	true I false

NN4.0
NN4.06
IE4.0
IE5.0

JavaScript

ディスプレイのサイズを取得する

screen.width [プロパティ]
screen.height [プロパティ]
screen.availWidth [プロパティ]
screen.availHeight [プロパティ]

*ディスプレイのサイズを取得する

スクリーンの幅(pixels):1024
スクリーンの高さ(pixels):768
使用可能なスクリーンの幅(pixels):1024
使用可能なスクリーンの高さ(pixels):740

解説

「width」プロパティと「height」プロパティは、スクリーン全体の幅と高さの値を返します。「availWidth」プロパティと「availHeight」プロパティは、スクリーン全体からWindowsのタスクバーのように、表示できない部分を除いたスクリーンの幅と高さの値をそれぞれ返します。
JavaScript1.2で追加されたプロパティです。これらのプロパティは読み出し専用です。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ディスプレイのサイズを取得する<P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
document.write("スクリーンの幅(pixels):",screen.width);
document.write("<BR>");
document.write("スクリーンの高さ(pixels):",screen.height);
document.write("<BR>");
document.write("使用可能なスクリーンの幅(pixels):",screen.avail
Width);
document.write("<BR>");
document.write("使用可能なスクリーンの高さ(pixels):",screen.avail
Height);
//--->
</SCRIPT>
</BODY>
</HTML>
```

ディスプレイ情報を利用する

ディスプレイの表示情報を取得する

screen.pixelDepth [プロパティ]
screen.colorDepth [プロパティ]

*ディスプレイの表示情報を取得する

ディスプレイ深度(bits per pixel):32
使用可能カラー数(bitカラー):32

解説

「pixelDepth」プロパティは1ピクセルあたり何ビットの情報で表示するかの値を、「colorDepth」プロパティは表示可能色数の値を、ビット数でそれぞれ返します。例えば、256色の場合は8、65000色(Macintoshでは32000色)の場合は16となります。JavaScript1.2で追加されたプロパティです。これらのプロパティは読み出し専用です。Internet Explorerでは、「pixelDepth」に未対応です。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ディスプレイの表示情報を取得する<P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write(" ディスプレイ深度(bits per pixel):",screen.
pixelDepth);
document.write("<BR>");
document.write(" 使用可能¥カラー数(bit カラー):",screen.color
Depth);
//-->
</SCRIPT>
</BODY>
</HTML>
```

イベントのタイプを取得する

JavaScript

event.type	←イベントの種類	[プロパティ]
onClick		[イベント]
onMouseOut		[イベント]
onMouseDown		[イベント]
onMouseUp		[イベント]
onMouseOver		[イベント]

*イベントのタイプを取得する

onClick Event!!

onMouseOut Event!!

onMouseDown Event!!

onMouseUp Event!!

onMouseOver Event!!

Microsoft Internet Explorer
イベントタイプ:click
OK

Microsoft Internet Explorer
イベントタイプ:mouseout
OK

Microsoft Internet Explorer
イベントタイプ:mousedown
OK

Microsoft Internet Explorer
イベントタイプ:mouseup
OK

Microsoft Internet Explorer
イベントタイプ:mouseover
OK

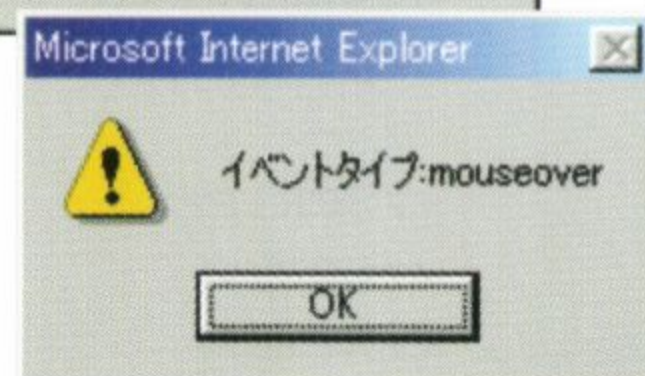
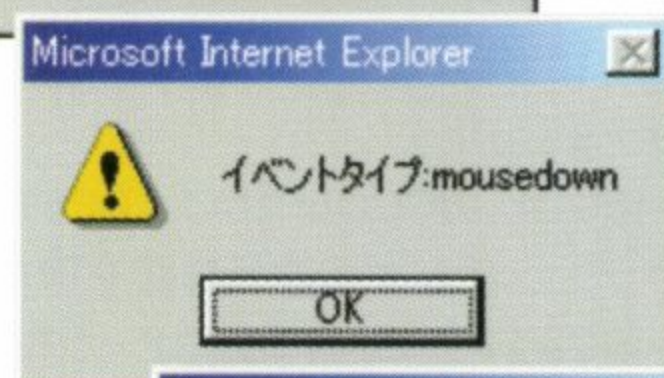
*イベントのタイプを取得する

onClick Event!!

onMouseOut Event!!

onMouseDown Event!!

onMouseUp Event!!

onMouseOver Event!!

解説

「type」プロパティは、発生したイベントのタイプの値を持っています。
サンプルでは、フォームやリンクに色々なイベントハンドラを設定し、イベントが発生した時に警告用のダイアログボックスにそのイベントタイプを表示しています。
「onClick」はオブジェクトがクリックされた時、「onMouseOut」はオブジェクトからマウスカーソルが離れた時、「onMouseDown」はマウスボタンが押し下げられた時、「onMouseUp」はマウスボタンが離された時、「onMouseOver」はオブジェクト上にマウスポインタが乗った時をそれぞれイベントとして捉えます。
JavaScript1.2で追加されたプロパティです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*イベントのタイプを取得する<P>
<FORM NAME= "EVENT1">
 <INPUT TYPE="button" NAME= "event1" VALUE="onClick Event!
!"
 onClick="alert('イベントタイプ:'+ event.type)">
</FORM>
<A HREF="#" onMouseOut="alert('イベントタイプ:'+ event.type)">
onMouseOut Event!!
</A>
<P>
<A HREF="#" onMouseDown="alert('イベントタイプ:'+ event.type)">
onMouseDown Event!!
</A>
<P>
<A HREF="#" onMouseUp="alert('イベントタイプ:'+ event.type)">
onMouseUp Event!!
</A>
<P>
<A HREF="#" onMouseOver="alert('イベントタイプ:'+ event.type)">
onMouseOver Event!!
</A>
</BODY>
</HTML>
```

リファレンス「イベントタイプ」:P.550参照

JavaScript

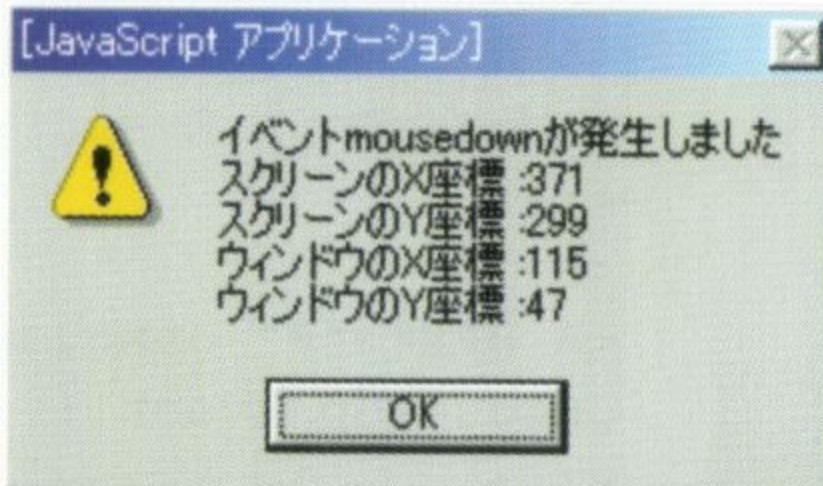
どこでイベントが発生したかを取得する

イベント.type	←イベントの種類	[プロパティ]
イベント.screenX	←ディスプレイのX軸の値	[プロパティ]
イベント.screenY	←ディスプレイのY軸の値	[プロパティ]
イベント.pageX	←ウィンドウのX軸の値	[プロパティ]
イベント.pageY	←ウィンドウのY軸の値	[プロパティ]
onMouseDown		[イベント]

Before

*どこでイベントが発生したかを取得する

After

[JavaScript アプリケーション]
イベントmousedownが発生しました
スクリーンのX座標 :371
スクリーンのY座標 :299
ウィンドウのX座標 :115
ウィンドウのY座標 :47
OK

解説

「screenX」と「screenY」プロパティはイベントが起こったディスプレイ上のX軸とY軸の値を、「pageX」と「pageY」プロパティはイベントが起こったウィンドウの表示領域上のX軸とY軸の値をそれぞれ持っています。
サンプルでは、マウスボタンが押された時に、イベントタイプ「document.onmousedown」によってイベントを取得して関数「eve1()」を発生させ、関数の処理で発生したイベントが持っているプロパティの値を取り出し、それを警告用のダイアログボックスに表示しています。
JavaScript1.2で追加された用法です。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function eve1(e) {
    alert (    "イベント"+e.type +"が発生しました" +   "¥n" +
 "スクリーンのX座標 :" + e.screenX + "¥n" +
 "スクリーンのY座標 :" + e.screenY + "¥n" +
 "ウィンドウのX座標 :" + e.pageX + "¥n" +
 "ウィンドウのY座標 :" + e.pageY );
   return true;
}
document.onmousedown = eve1;
//-->
```

```
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*どこでイベントが発生したかを取得する<P>
</BODY>
</HTML>
```

onMouseDown→リファレンス「イベントタイプ」の「MouseDown」:P.551参照

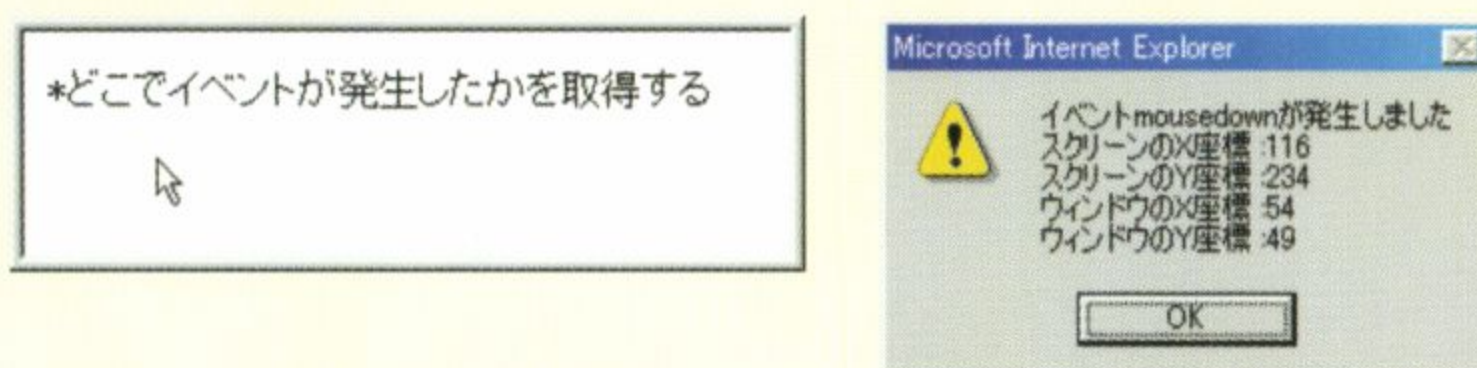
Tip Internet Explorerでのeventオブジェクトの使用法

Internet ExplorerのサポートしているJScriptでも、Internet Explorer4.0からNetscape Navigatorと同様にウィンドウ上のどこでイベントが発生しても、eventオブジェクトを使用して、そのイベントを取得できます。しかし、オブジェクトの取り扱い方などに、Netscape Navigatorと微妙な違いがあります。

取得するイベントを設定する方法は、Netscape Navigatorと同じです。しかし、Netscape Navigatorのeventオブジェクトが、直接イベントからイベントに関する情報を取り出すのに対して、Internet Explorerのeventオブジェクトは、windowオブジェクトのプロパティなので、「window.event.プロパティ」の用法でイベントの情報を取得することになります。

次のサンプルは、eventオブジェクトの「どこでイベントが発生したかを取得する」をInternet Explorerで使用できるようにしたものです。eventオブジェクトの設定方法の違いに注目してください。また、プロパティにも「pageX」や「pageY」のようにNetscape Navigatorのみでサポートされているものや、「offsetX」や「offsetY」のようにInternet Explorerのみでサポートされているものがあるので注意してください。

*どこでイベントが発生したかを取得する

Microsoft Internet Explorer
イベントmousedownが発生しました
スクリーンのX座標 :116
スクリーンのY座標 :234
ウィンドウのX座標 :54
ウィンドウのY座標 :49
OK

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function eve1() {
    alert (      "イベント"+window.event.type +"が発生しました" +  "¥n" +
 "スクリーンのX座標 :" + window.event.screenX + "¥n" +
 "スクリーンのY座標 :" + window.event.screenY + "¥n" +
 "ウィンドウのX座標 :" + window.event.offsetX + "¥n" +
 "ウィンドウのY座標 :" + window.event.offsetY );
   return true;
}
document.onmousedown = eve1;
//-->
</SCRIPT>
```

NN4.0
NN4.06

レイヤー上のどこでイベントが発生したかを取得する

JavaScript

イベント.layerX	←レイヤー上のX軸の値	[プロパティ]
イベント.layerY	←レイヤー上のY軸の値	[プロパティ]
onMouseDown		[イベント]

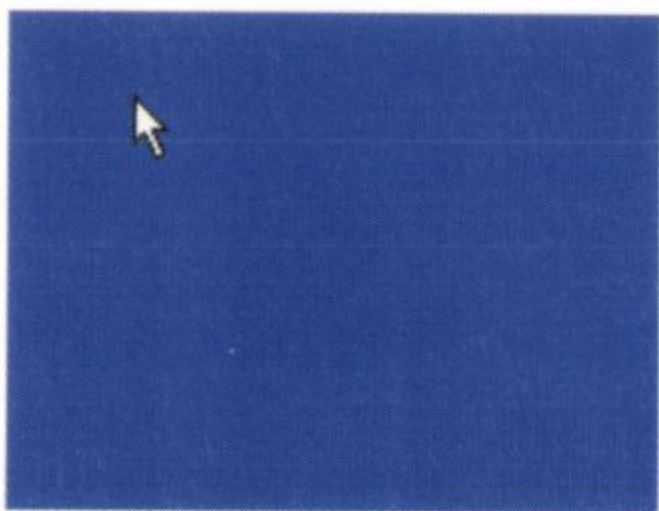
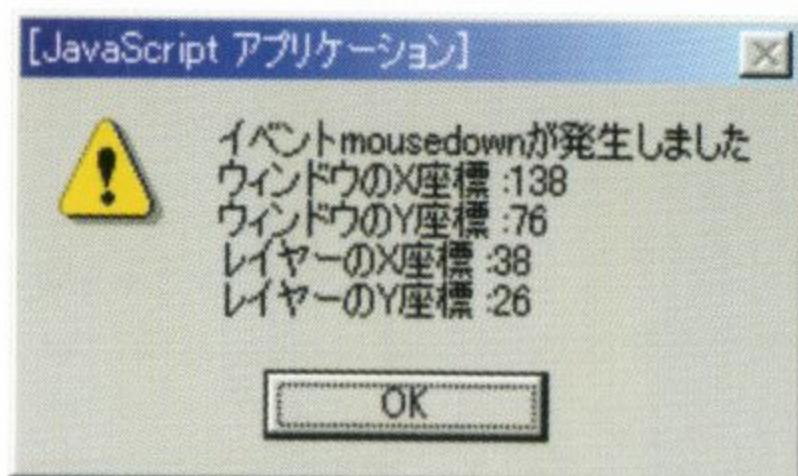
Before

*レイヤー上のどこでイベントが発生したかを取得する

After

[JavaScript アプリケーション]

イベントmousedownが発生しました
ウィンドウのX座標 :138
ウィンドウのY座標 :76
レイヤーのX座標 :38
レイヤーのY座標 :26

OK

解説

「layerX」と「layerY」プロパティは、イベントが起こったレイヤー上のX軸とY軸の値を持っています。
サンプルでは、レイヤー上でマウスボタンが押された時に、イベントタイプ「document.onmousedown」によってイベントを取得して関数「eve1()」を発生させ、イベントが持っているマウスカーソルのウィンドウとレイヤー上の位置の値を、警告用のダイアログボックスに表示しています。
JavaScript1.2で追加された用法です。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*レイヤー上のどこでイベントが発生したかを取得する<P>
<LAYER NAME="LAY1" VISIBILITY="show" BGCOLOR="blue" TOP
=50 LEFT=100 WIDTH=200 HEIGHT=150>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function eve1(e) {
    alert (     "イベント"+e.type +"が発生しました" +  "¥n" +
 "ウィンドウのX座標 :" + e.pageX + "¥n" +
 "ウィンドウのY座標 :" + e.pageY + "¥n" +
 "レイヤーのX座標 :" + e.layerX + "¥n" +
 "レイヤーのY座標 :" + e.layerY);
   return true;
}
document.onmousedown = eve1;
//-->
</SCRIPT>
</LAYER>
</BODY>
</HTML>
```

onMouseDown→リファレンス「イベントタイプ」の「MouseDown」:P.551参照

どのキーが押されたかを取得する

イベント.modifiers　[プロパティ]
イベント.which　[プロパティ]
onkeydown　[イベント]

Before

*どのキーが押されたかを取得する

After

[JavaScript アプリケーション]
jkey が押されました
OK

解説

「which」プロパティは、イベント発生時に押されたキーのASCIIコードの値を持っています。
サンプルでは、キーが押されたことを「document.onkeydown」によってイベントとして捉え、「which」の値を警告用のダイアログボックスに表示しています。
その時、ASCIIコードの値そのままでは、「a」のキーは「65」、「b」のキーは「66」といったコード番号なので、それをstringオブジェクトの「fromCharCode()」メソッドを使用して、英数字に変換しています。
JavaScript1.2で追加された用法です。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function eve1(e) {
    alert ( String.fromCharCode(e.which) + ":key が押されました") ;
    return true ;
}
document.onkeydown = eve1 ;
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*どのキーが押されたかを取得する<P>
</BODY>
</HTML>
```

String.fromCharCode()→「stringオブジェクト」の「ISO-Latin-1のコード番号を文字に変換する」:P.493参照

JavaScript

ドラッグ＆ドロップしていいかを確認する

NN4.0
NN4.06

イベント.data ［プロパティ］
onDragDrop ［イベント］

Before

*ドラッグ&ドロップしていいかを確認する

ウィンドウを開く

*元のウインドウにドラッグ&ドロップしてください

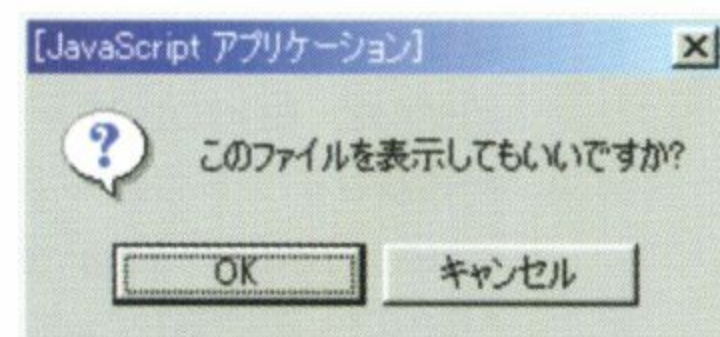
image.gif

Netscape社のHomePageへ...

After

[JavaScript アプリケーション]

このファイルを表示してもいいですか?

OK　キャンセル

解説

「onDragDrop」イベントは、ウィンドウ上で、画像やHTMLファイルなどのドラッグ＆ドロップの動作が行われた時のイベントを取得します。
また、「data」プロパティは、ドロップされた画像やHTMLファイルなどのURLの値を持っています。
サンプルでは、画像やリンクをウィンドウ上にドロップした時に、「window.onDragDrop」によってイベントを取得して関数「eve1()」を発生させ、関数の処理でファイルを表示していいかどうかを確認するダイアログボックスを開いています。
ただし、Windows95版のNetscape Navigator4.xは、画像のドラッグ＆ドロップに対応していません。
JavaScript1.2で追加された用法です。

NN4.0
NN4.06

【ドラッグ&ドロップされるウィンドウ】

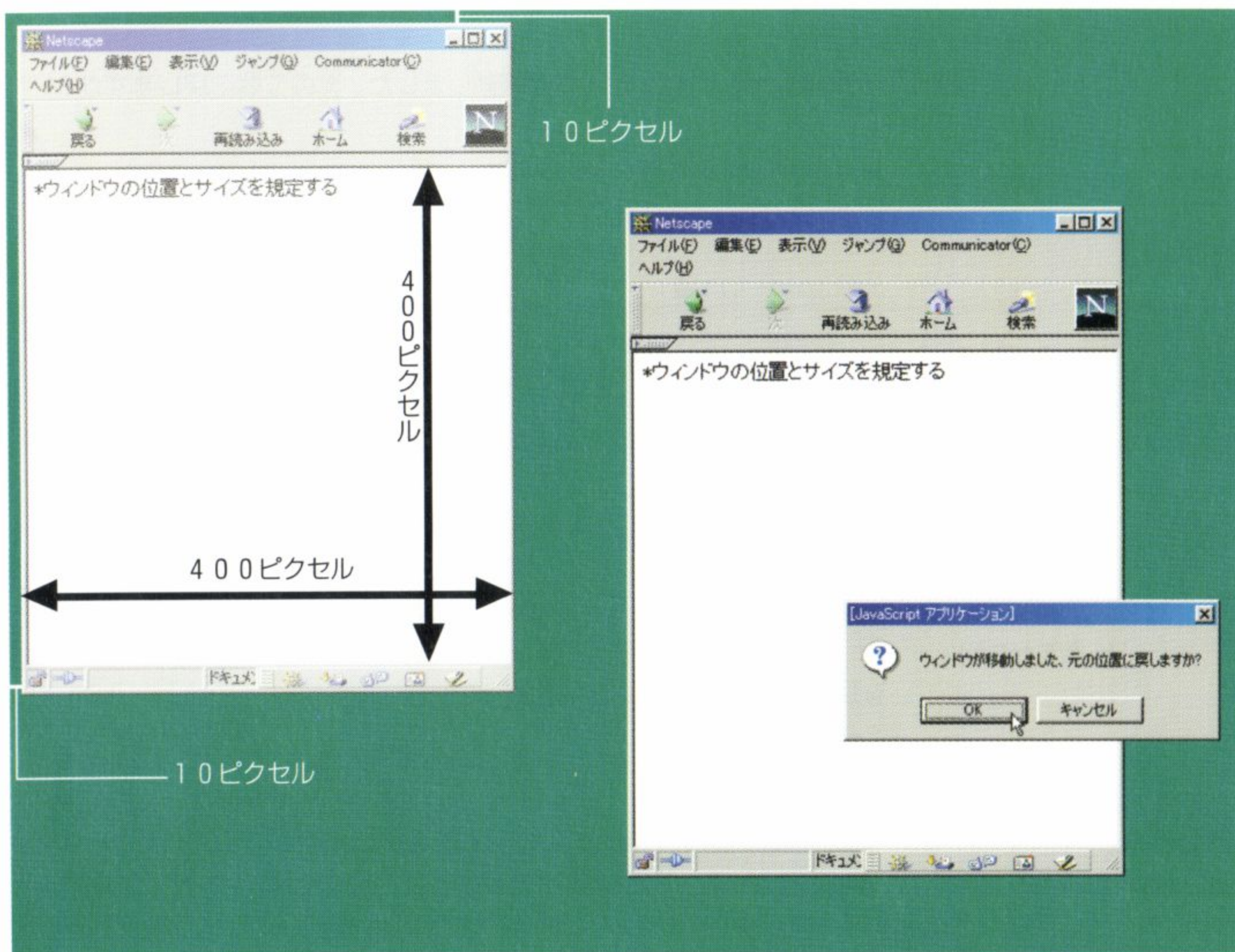
```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function eve1() {
    if ( confirm ( "このファイルを表¥示してもいいですか?") ) {
        return true;
        }
    return false
}
window.onDragDrop = eve1;
//-->
</SCRIPT>
</HEAD><BODY BGCOLOR="#FFFFFF">
*ドラッグ&ドロップしていいかを確認する<P>
<A HREF="NewWin.html" TARGET="_Open">ウィンドウを開く</A>
</BODY>
</HTML>
```

【ドラッグ&ドロップ元のウィンドウ】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*元のウインドウにドラッグ&ドロップしてください<P>
<IMG SRC="image.gif" NAME="eve05" ALT="image.gif" WIDTH=
"100" HEIGHT="100"><P>
<A HREF="http://www.netscape.com/">Netscape 社の HomePage
へ...</A>
</BODY>
</HTML>
```

onDragDrop→リファレンス「イベントタイプ」の「DragDrop」:P.550参照

ウィンドウの位置とサイズを規定する

onMove 【イベント】
onResize 【イベント】

解説

イベント「onMove」はウィンドウが移動した時、「onResize」はウィンドウのサイズが変更された時のイベントを取得します。

サンプルでは、HTMLファイルが読み込まれた時にウィンドウのディスプレイ上の表示位置とサイズを確定しています。そして「window.onMove」「window.onResize」によって、それ以降ウィンドウが動かされたり、ウィンドウサイズが変更された時をイベントとして取得します。ウィンドウの位置やサイズが意図したものでなかった場合、元に戻すか確認した後、ウィンドウの位置やサイズを戻しています。ウィンドウの位置は、1度だけ元の位置に戻さないようにすれば、それ以降は自由に位置を変更できるようになります。

1度のウィンドウの移動で、複数回の「onMove」イベントが発生することがあります。サンプルでは、1番初めに発生したイベントについてのみ、処理を行うようにしています。また、Windows版のNetscape Navigatorでは、スクロールバーが表示された時にも「onResize」イベントが発生するので、注意してください。

JavaScript1.2で追加された用法です。

NN4.0
NN4.06

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
window.moveTo(10,10);
window.resizeTo(400,400);
RL=1
function eve1() {
    if(RL==1){
        if ( confirm ("ウィンドウが移動しました、元の位置に戻しますか?"
) ) {
            window.moveTo(10,10);
            location.reload();
            RL++;
            return true;
        }
        else { RL=2 }
    }
}
function eve2() {
    if (window.outerHeight <= 380 || window.outerWidth
<= 380 ){
        if ( confirm ("ウィンドウのサイズが小さすぎます、元のサイズに戻
しますか?") ) {
            window.resizeTo(400,400);
            return true;
        }
        return false;
    }
}
window.onMove = eve1;
window.onResize = eve2;
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ウィンドウの位置とサイズを規定する<P>
</BODY>
</HTML>
```


outerHeight→「windowオブジェクト」の「ウィンドウの外周･内周を取得する」:P.307参照

outerWidth→「windowオブジェクト」の「ウィンドウの外周･内周を取得する」:P.307参照

moveTo()→「windowオブジェクト」の「ブラウザを指定した位置へ移動する」:P.308参照

resizeTo()→「windowオブジェクト」の「ブラウザの大きさを指定してリサイズする」:P.310参照

location.reload()→「locationオブジェクト」の「リロードボタンを作る」:P.360参照

onMove→リファレンス「イベントタイプ」の「Move」:P.552参照

onResize→リファレンス「イベントタイプ」の「Resize」:P.552参照

警告用のダイアログボックスを開く

alert(文字列) 【メソッド】
onClick 【イベントハンドラ】

Before

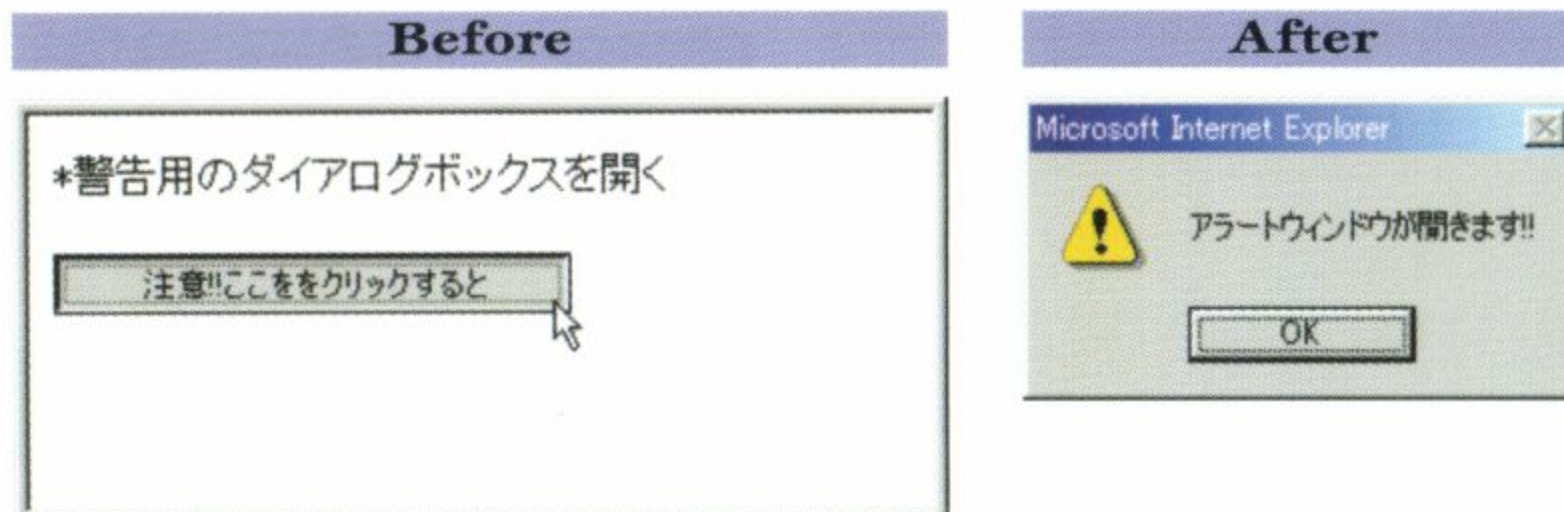
*警告用のダイアログボックスを開く

注意!!ここをクリックすると

After

Microsoft Internet Explorer

アラートウィンドウが開きます!!

OK

解説

「alert()」は、警告用のダイアログボックスを開くメソッドです。
サンプルでは、ボタンをクリックした時にイベントハンドラ「onClick」が関数「EVENT1()」を発生させ、ダイアログボックスを開いています。
利用者に注意をうながす時などに使用します。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function EVENT1(){ window.alert("アラートウィンドウが開きます!!") }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*警告用のダイアログボックスを開く<P>
<FORM>
 <INPUT TYPE="button" NAME= "CF" VALUE=" 注意!!ここをクリックすると" onClick="EVENT1()">
</FORM>
</BODY></HTML>
```

注意

alert()を使いすぎていませんか?

「alert()」で開くダイアログボックスは、その名の通り、利用者に注意をうながすためのダイアログボックスです。そのため、このダイアログボックスが開く時は警告音を発し、開いている間は他の作業を受け付けなくなります。

割と短いコードで明確な効果が得られるので、JavaScriptの解説ではよく使われていますが、実際に利用する時はその用途に合っているかどうかを十分考慮した上で利用してください。

確認ボタン付きのダイアログボックスを開く

JavaScript

confirm(文字列) [メソッド]

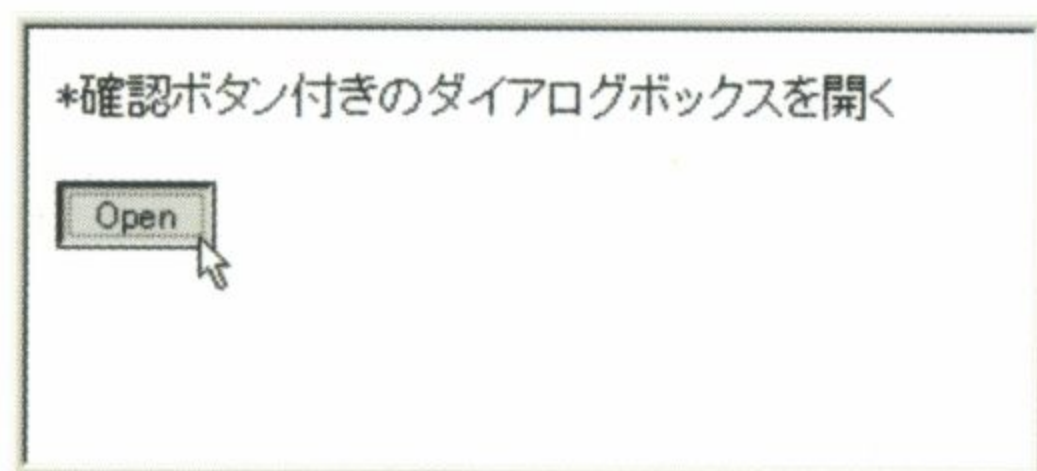
Before

*確認ボタン付きのダイアログボックスを開く

Open

After

Microsoft Internet Explorer

OK(Yes).Cancel(No)ボタン付きのダイアログボックス

OK キャンセル

解説

「confirm()」は、確認用ボタン付きのダイアログボックスを開くメソッドです。
確認ボタンの名称は、Windows版のNetscape Navigatorは「OK/キャンセル」、Macintosh版のNetscape Navigatorは「Yes/No」と、OSによって違います。
「OK」、又は「Yes」のボタンが押された時は真(true)の値を、「キャンセル」、又は「No」のボタンが押された時は偽(false)の値を返します。
サンプルでは、押されたボタンによって違うページがロードされるようになっています。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function EVENT3(){
    if ( confirm ("OK(Yes).Cancel(No)ボタン付ダイアログボックス"
) ) { location.href="wp1.html" }
    else { location.href="wp2.html" }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*確認ボタン付きのダイアログボックスを開く<P>
<FORM>
 <INPUT TYPE="button" NAME= "CF" VALUE=" Open " onClick=
"EVENT3()">
</FORM>
</BODY></HTML>
```

location.href→「locationオブジェクト」の「自ページのURLを取得する」:P.353参照

入力欄付きのダイアログボックスを開く

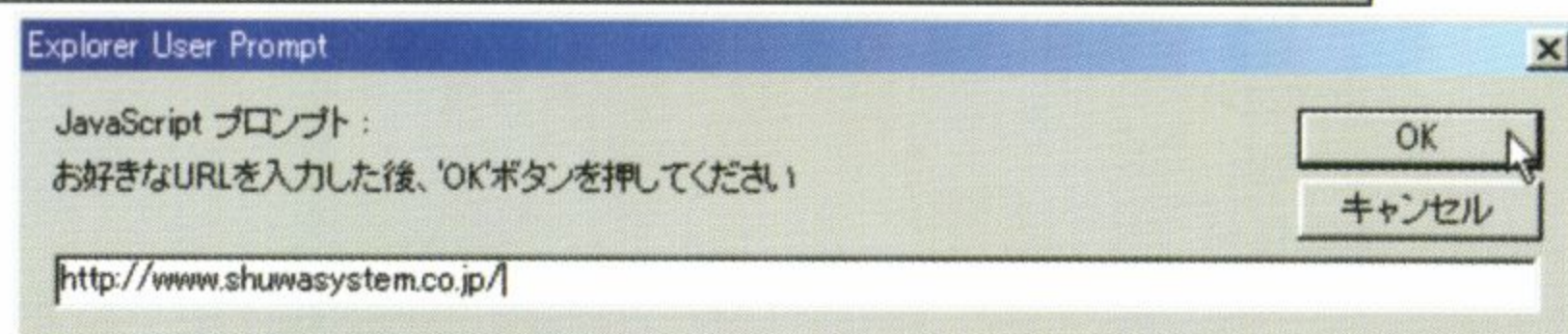
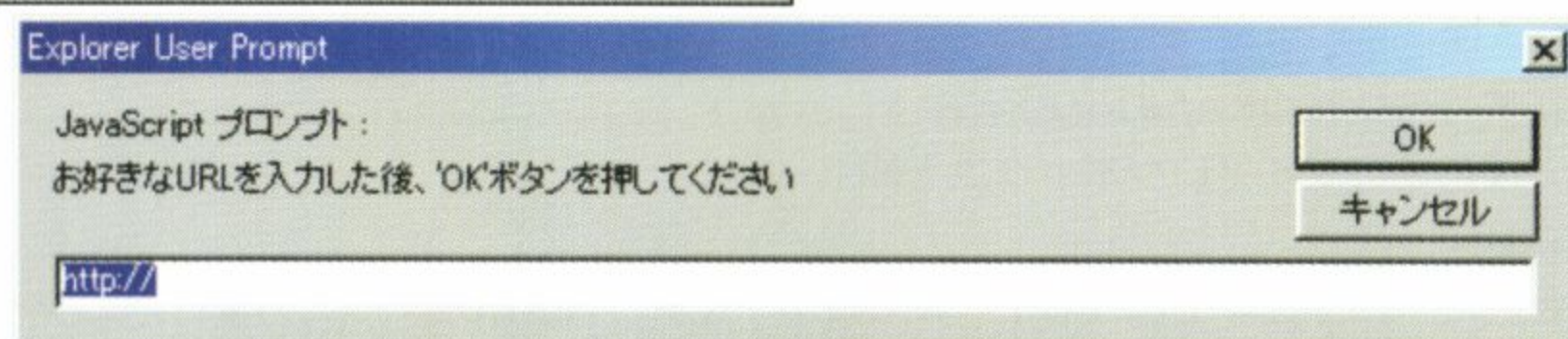
prompt(文字列,値) [メソッド]

*入力欄付きのダイアログボックスを開く

入力欄付きのダイアログボックス

Explorer User Prompt
JavaScript プロンプト：
お好きなURLを入力した後、'OK'ボタンを押してください
OK
キャンセル
http://

Explorer User Prompt
JavaScript プロンプト：
お好きなURLを入力した後、'OK'ボタンを押してください
OK
キャンセル
http://www.shuwasystem.co.jp/

Shuwa System
Contents
▼書籍情報 99/07/24
├新刊書籍 NEW!
├既刊書検索
├ダウンロード
├LinuxJapan NEW!
├フェア開催
├特集 工事中
└書籍購入法
▼ゲーム情報 随時更新
▼ソフト情報 99/01/11
▼リクルート情報 99/06/15
Excel2000VBA 実践プログラミング

解説

「prompt()」は、入力欄付きのダイアログボックスを開くメソッドです。「prompt(ウィンドウ内に表示する文字列,入力ボックス内の初期値)」と指定します。「OK」ボタンが押されると入力欄付きダイアログボックス内の値が代入され、キャンセルボタンが押されるとnullの値を返します。
サンプルでは、入力欄に何も入力されていなかったり、初期値のままだったり、キャンセルボタンが押された時には「alert()」で警告用のダイアログボックスを開き、それ以外の場合は入力されたURLをロードします。もし、その時に入力されたURLが不正な場合は、ブラウザ自身が警告をします。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function EVENT4(){
    PRO=prompt("お好きなURLを入力した後、'OK'ボタンを押してください",
"http://")
        if (!(PRO=="" || PRO==null || PRO=="http://"))
{ location.href=PRO }
        else { alert("何も入力されていないか、Cancelが押されました") }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*入力欄付きのダイアログボックスを開く<P>
<FORM>
 <INPUT TYPE="button" NAME= "CF" VALUE=" 入力欄付きのダイアログ
ボックス" onClick="EVENT4()">
</FORM>
</BODY>
</HTML>
```

location.href→「locationオブジェクト」の「自ページのURLを取得する」:P.353参照

JavaScript

ステータス行にメッセージを表示する

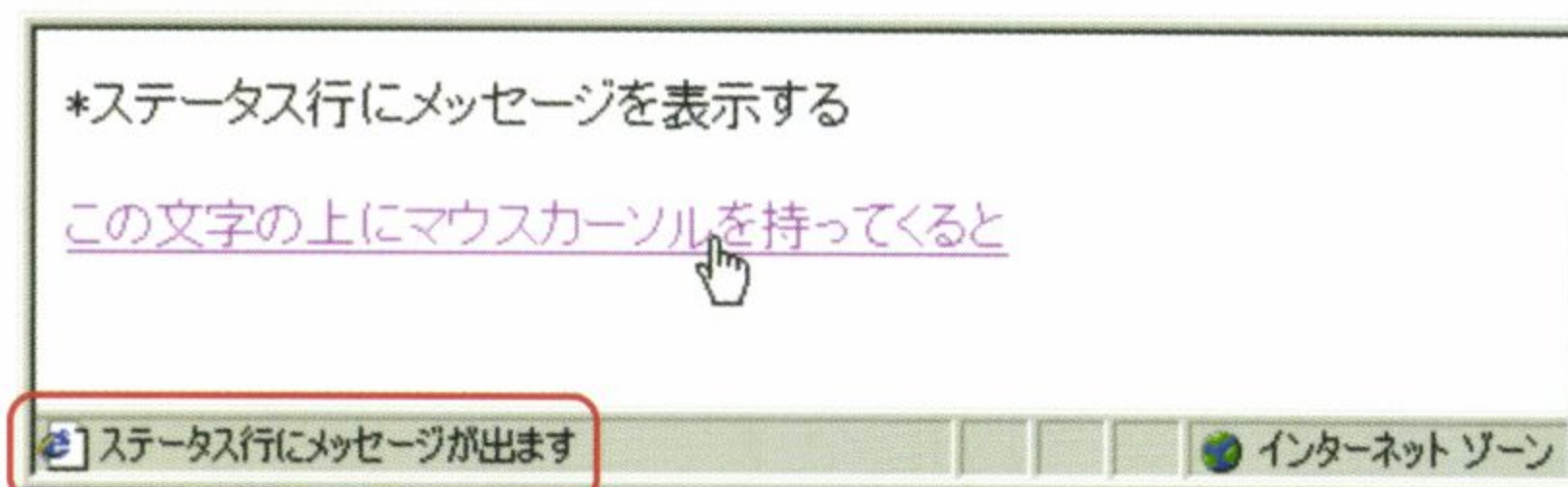
window.status [プロパティ]
onMouseOver [イベントハンドラ]

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

JavaScript

*ステータス行にメッセージを表示する

この文字の上にマウスカーソルを持ってくると

ステータス行にメッセージが出ます　インターネット ゾーン

解説

サンプルでは、リンク上にマウスポインタが乗った時にイベントハンドラ「onMouse Over」内のスクリプトが評価され、「status」プロパティによってステータス行に文字を表示させています。
リンクの説明を表示するなどの利用方法があります。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ステータス行にメッセージを表示する<P>
<A HREF="#"onMouseOver="window.status='ステータス行にメッセージ
が出ます';return true">この文字の上にマウスカーソルを持ってくると</A>
</BODY>
</HTML>
```

onMouseOver→リファレンス「イベントハンドラ」の「onMouseOver」:P.549参照

JavaScript

ステータス行に文字を流す

window.status [プロパティ]
setTimeout(*処理,時間設定*) [メソッド]
clearTimeout() [メソッド]

*ステータス行に文字を流す

ここにメッセージを入れます.....　インターネット ゾーン

ここにメッセージを入れます.....。

ここにメッセージを入れます.....。

解説

サンプルは、まずページが読み込まれた時に、イベントハンドラ「onLoad」が関数「Mess()」を発生します。
次に「Mess(){...}」内では、TCの値に1を加え、「window.status」でステータス行に文字列を書き出した後、stringオブジェクトの「substring()」で文字列を2字ずらし、「setTimeout()」で再び関数「Mess()」を呼び出しています。そして、この処理を「(TC<1000)」が真になるまで繰り返すことによって、ステータス行に文字が流れるような効果を出しています。
「Mess()」の処理が終った時は、ステータス行にスペースを表示することによって、ステータス行の文字をクリアしています。
文字列の最後と先頭がくっついたり、いきなり文字が現われたりしないように、文字列の始めにスペースを入れています。
スクロールされている長さの調整は、「(TC < 1000)」内の数値を変更することによって行います。
スクロールの速度は、「setTimeout("Mess()",300)」内の数値を変えることによってミリ秒単位で調整します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
var TC = 0 ;
var Sm1 = "                                          ";
var Sm2 = "                                          ";
var Sm3 = "                                          ";
```

```
var Sm4 = "ここにメッセージを入れます......。";
var Smess = Sm1+Sm2+Sm3+Sm4;
var timer=0 ;
function Mess() {
    if (TC < 1000) { //ここの数値を変えることによってスクロールする時間
が変わります
        TC++ ;
            window.status = Smess;
            Smess = Smess.substring(2,Smess.length) +
Smess.substring(0,2);
            var timeID=setTimeout("",1) ;
            clearTimeout(timeID) ;
            timeID=timer = setTimeout("Mess()",300);
    }
    else { window.status = " " }
}
//--->
</SCRIPT>
</HEAD>
<BODY  BGCOLOR="#FFFFFF" onLoad="Mess()">
*ステータス行に文字を流す
</BODY>
</HTML>
```

substring()→「stringオブジェクト」の「文字列の途中の文字を抜き出す」:P.488参照

onLoad→リファレンス「イベントハンドラ」の「onLoad」:P.548参照

注意

「setTimeout()」の処理を繰り返した時の注意

Netscape Navigator2.0でsetTimeout()の処理を繰り返したJavaScriptを実行した時、しばらくすると「メモリーエラー」が発生し、ブラウザの動作が不安定になってしまいます。

本書ではこのエラーを回避するため、一定時間で処理を終了させる、という対策をとっています。

その他にも、エラー回避のために"clearTimeout()"を使って、1回1回「setTimeout()」の処理を明示的に終了させる、などの対策があります。

このエラーはNetscape Navigator3.0以上では発生しません。

JavaScript

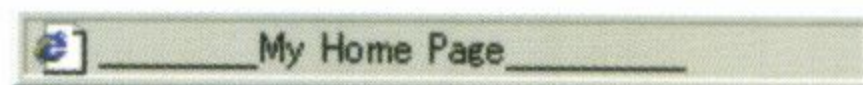
ページがロードされた時にステータス行に挨拶を表示する

window.status [プロパティ]
setTimeout(*処理,時間設定*) [メソッド]

*ページがロードされた時にステータス行に挨拶を表示する

_______Welcome__________ インターネット ゾーン

_______To__________

_______My Home Page_________

解説

サンプルでは、ステータス行に表示する文字列の配列を作り、ページが読み込まれた時に、配列の要素を順番にステータス行に書き出しています。
表示の処理が終った時には、ステータス行にスペースを表示することによって、ステータス行の文字をクリアしています。
表示のタイミングは、「setTimeout("EVENT3()", 500)」内の数値を変えることによってミリ秒単位で調整します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
var TC = 0 ;
var j = 0
function MakeArray(n) {
   this.length = n ;
   for (var i = 0; i <= n; i++) {
      this[i] = 0;
   }
   return this ;
}
msg = new MakeArray(16) ;
msg[0] = " " ;
msg[1] = " " ;
msg[2] = "______________Welcome_________________" ;
msg[3] = "______________Welcome_________________" ;
msg[4] = "______________________________________" ;
msg[5] = "_________________To___________________" ;
msg[6] = "_________________To___________________" ;
```

```
msg[7] = "________________________________________________" ;
msg[8] = "________________My Home Page_________________" ;
msg[9] = "________________My Home Page_________________" ;
msg[10] = "                                              " ;
msg[11] = "                                              " ;
msg[12] = "                                              " ;
msg[13] = "______*Welcome To My Home Page*_________" ;
msg[14] = "______*Welcome To My Home Page*_________" ;
msg[15] = "______*Welcome To My Home Page*_________" ;
function EVENT3() {
   if (TC < 16) {
      TC++ ;
         if (j <= msg.length) {
         window.status = msg[j] ;
         j++ ;
         if (j == msg.length) {
            j = 0 ;
         }
         setTimeout("EVENT3()", 500);
      }
   } else {
     window.status = " " ;
   }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF" onLoad="EVENT3()">
*ページがロードされた時にステータス行に挨拶を表示する
</BODY>
</HTML>
```

length→「複数のオブジェクトで使用するプロパティ・メソッド」の「オブジェクト(配列)の数を取得する」:P.512参照

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

JavaScript

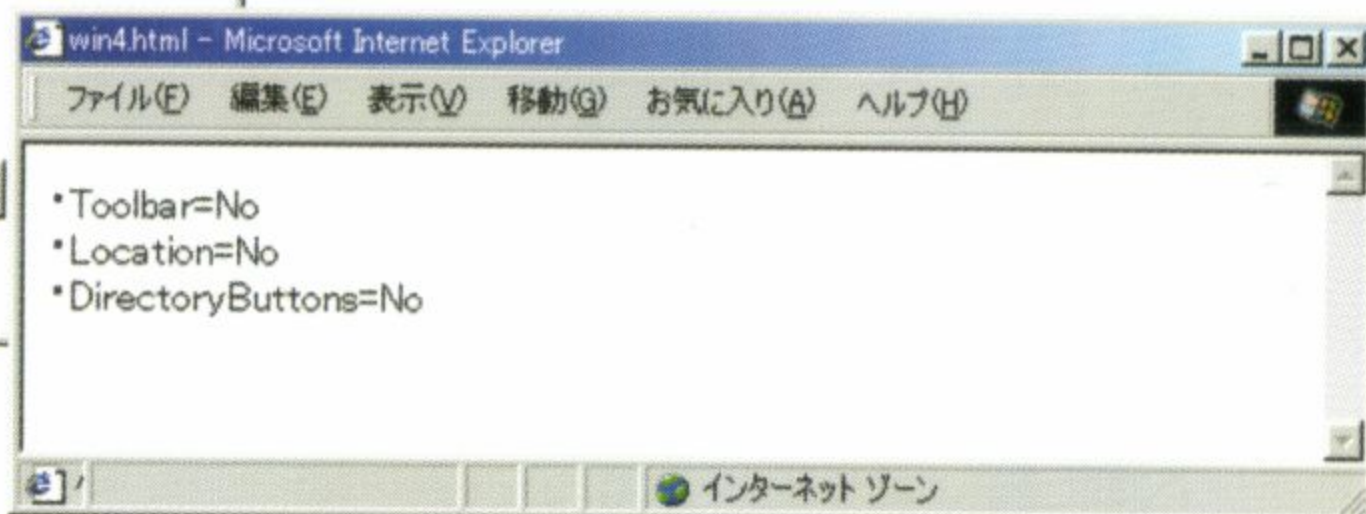
新しいウィンドウを開く

window.open("*URL*","*ウィンドウ名*","*属性*") [メソッド]

*新しいウィンドウを開く(Mac&X版Netscape2.x以外)

全部あり
デレクトリなし
デレクトリ・ロケーションなし
デレクトリ・ロケーション・ツールバーなし
サイズ指定あり

win4.html - Microsoft Internet Explorer

・Toolbar=No
・Location=No
・DirectoryButtons=No

解説

「open()」メソッドは、新しいウィンドウを開くメソッドです。
「URL」にはウィンドウ内に表示するHTMLファイルのURLを、「ウィンドウ名」には任意のウィンドウ名を指定します。「ウィンドウ名」を同じにすると、同じウィンドウとして取り扱われます。
「属性」の部分では、ウィンドウのツールバーやメニューバーなどの有無、ウィンドウのサイズなどを指定します。各項目は「,」で区切り、その項目が必要であれば「=yes」又は「=1」と指定し、不要であれば「=no」又は「=0」と指定し、ウィンドウのサイズは「=ピクセル」と指定します。指定できる属性は、「toolbar(ツールバー)」、「location(ロケーションバー)」、「directories(ディレクトリーバー)」、「status(ステータスバー)」、「menubar(メニューバー)」、「scrollbars(スクロールバー)」、「resizable(リサイズボックス)」、「width=pixels(ウィンドウの横幅)」、「height(ウィンドウの縦幅)」で、各項目は省略すると「yes」として判断されます。
Macintosh版のNetscape Navigator2.0など、一部のブラウザではURLが引けないものがあります。これらのブラウザも対象内に入れたWebページを作る場合は、右ページの「注意」を参照してください。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function wopen1(){ window.open("win1.html","WindowOpen1",
                    "toolbar=yes,location=yes,directories=
yes,status=yes,menubar=yes,scrollbars=yes,resizable=yes") }
function wopen2(){ window.open("win2.html","WindowOpen2",
                    "toolbar=yes,location=yes,directories=
no,status=yes,menubar=yes,scrollbars=yes,resizable=yes") }
function wopen3(){ window.open("win3.html","WindowOpen3",
                    "toolbar=yes,location=no,directories=
no,status=yes,menubar=yes,scrollbars=yes,resizable=yes") }
```

ウィンドウを操作する

```
function wopen4(){ window.open("win4.html","WindowOpen4",
                  "toolbar=no,location=no,directories=no,
status=yes,menubar=yes,scrollbars=yes,resizable=yes") }
function wopen5(){ window.open("win5.html","WindowOpen5",
                   "toolbar=no,location=no,directories=
no,status=yes,menubar=yes,scrollbars=yes,resizable=yes,
width=300,height=450") }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*新しいウィンドウを開く(Mac&X版Netscape2.x以外)<P>
<FORM>
 <INPUT TYPE="button" NAME= "Type1" VALUE="全部あり" onClick=
"wopen1()">
<BR>
 <INPUT TYPE="button" NAME= "Type2" VALUE="ディレクトリなし" on
Click="wopen2()">
<BR>
 <INPUT TYPE="button" NAME= "Type3" VALUE="ディレクトリ·ロケーショ
ンなし" onClick="wopen3()">
<BR>
 <INPUT TYPE="button" NAME= "Type4" VALUE="ディレクトリ·ロケーショ
ン·ツールバーなし" onClick="wopen4()">
<BR>
 <INPUT TYPE="button" NAME= "Type5" VALUE="サイズ指定あり" on
Click="wopen5()">
</FORM>
</BODY></HTML>
```

注意

Netscape Navigator2.xで「window.open()」メソッドを使用する時の注意

Machintosh版とUNIX版のNetscape Navigator2.xには、「window.open()」メソッドでURLが引けないという問題があります。そのため、通常のスクリプトの書き方では、開いたウィンドウに何も表示されません。

それらのブラウザを対象としたページで「window.open()」メソッドを使用する場合は、

```
function wopen1(){ var WO1;
                   WO1=window.open(" ","WindowOpen6",
                   "toolbar=yes,location=yes,directories
=yes,status=yes,menubar=yes,scrollbars=yes,resizable=yes");
                   WO1.location.href="URL" }
```

といった様にすれば大丈夫です。

この時、URLは「http://」から始まるフルパスで指定してください。

このスクリプトは、Machintosh版とUNIX版のNetscape Navigator2.x以外でも、正常に機能します。

JavaScript

ウィンドウを閉じる

window.close()	[メソッド]
setTimeout(*処理,時間設定*)	[メソッド]
onLoad	[イベントハンドラ]

Before

*ウィンドウを閉じる

始めにウィンドウを開きます

After

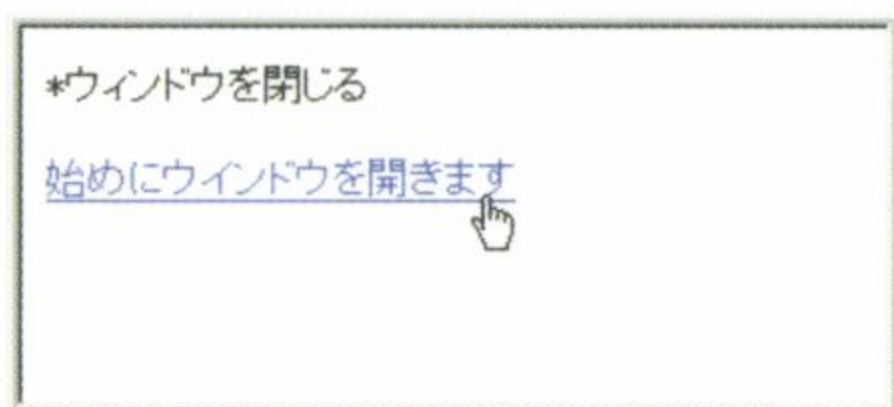
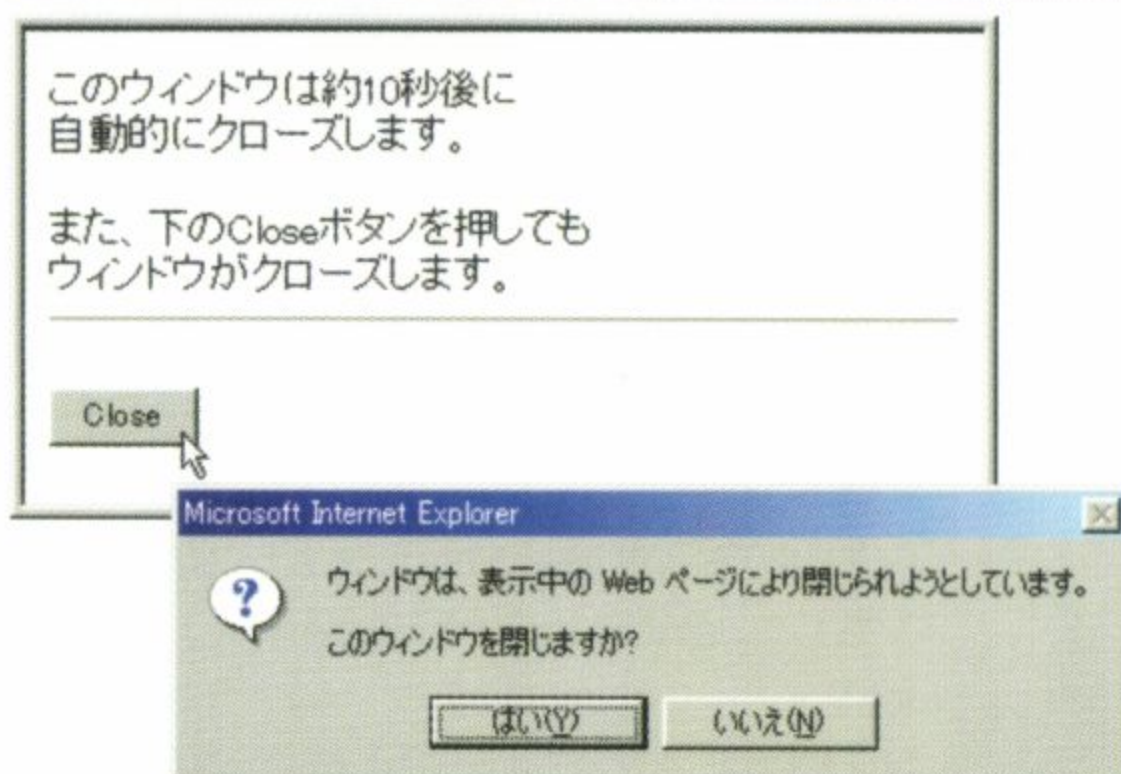
このウィンドウは約10秒後に
自動的にクローズします。

また、下のCloseボタンを押しても
ウィンドウがクローズします。

Close

Microsoft Internet Explorer

ウィンドウは、表示中の Web ページにより閉じられようとしています。
このウィンドウを閉じますか?

はい(Y)　いいえ(N)

解説

「close()」は、ウィンドウを閉じるメソッドです。
サンプルでは、ページがロードされた時に、<BODY>タグ内のイベントハンドラ「onLoad」が「setTimeout()」を呼び出し、10秒後に「window.close()」が発生してウィンドウが閉じます。また、フォームのボタンを押しても、イベントハンドラ「onClick」が「window.close()」を発生させ、ウィンドウが閉じます。
setTimeoutはミリ秒(1000分の1秒)単位で時間を設定できますが、あまり小さい数字を設定しても効果が出ません。
Netscape Navigator3.0からは、「close()」が発生した時にウィンドウが来歴情報を持っていれば、ウィンドウが閉じる時に確認のダイアログボックスを開くようになりました。

Sample

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF" onLoad="setTimeout('window.close
()',10000)">
このウィンドウは約10秒後に<BR>
自動的にクローズします。<P>
また、下のCloseボタンを押しても<BR>
ウィンドウがクローズします。
<HR>
<FORM>
 <INPUT TYPE="button" NAME= "ok" VALUE=" Close " onClick
="window.close()">
</FORM>
</BODY></HTML>
```

JavaScript

ページを抜ける時に新しいウィンドウを開く

window.open() [メソッド]
onUnload [イベントハンドラ]

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

Before

*ページを抜ける時に新しいウィンドウを開く

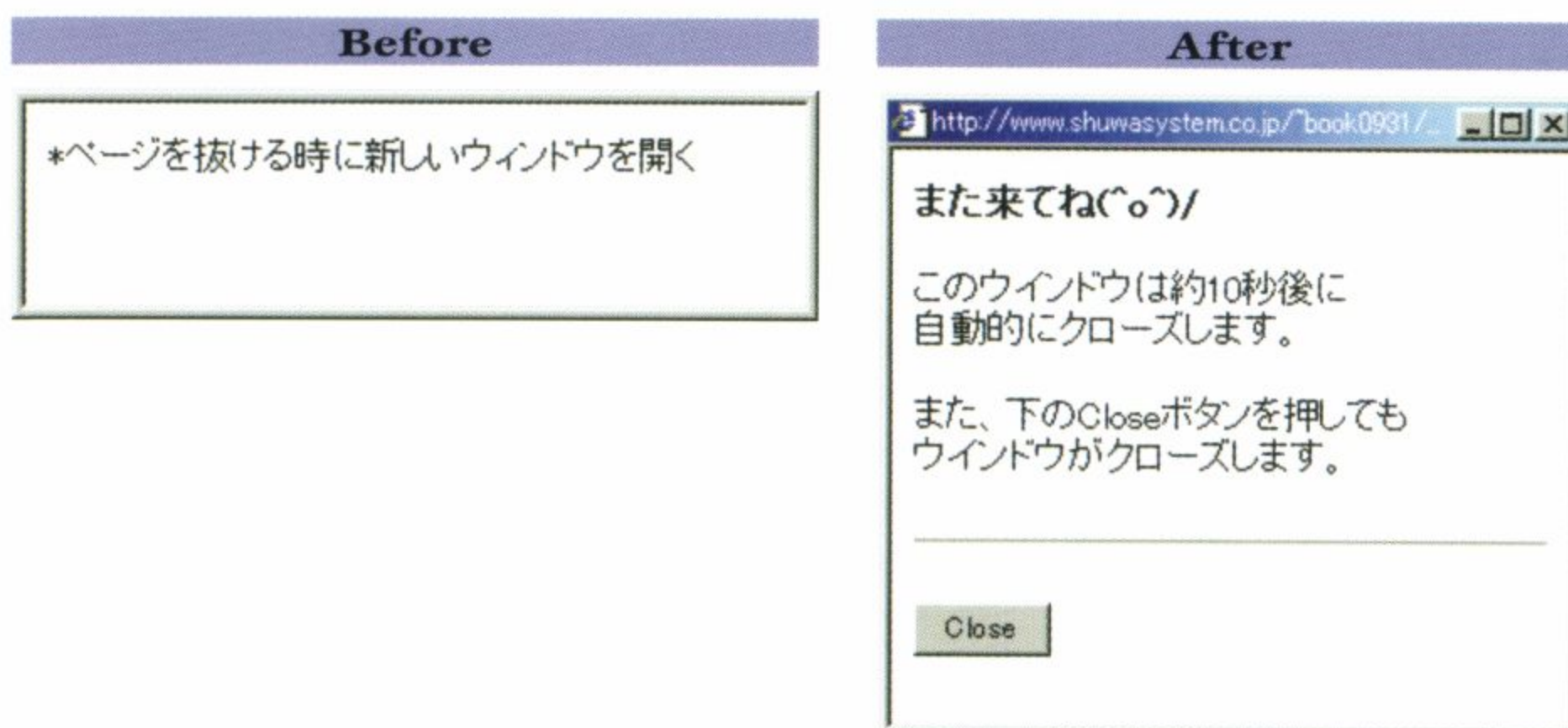
After

http://www.shuwasystem.co.jp/~book0931/...

また来てね(^o^)/

このウインドウは約10秒後に
自動的にクローズします。

また、下のCloseボタンを押しても
ウインドウがクローズします。

Close

解説

サンプルでは、別のページに抜ける時に、イベントハンドラ「onUnload」が関数を発生させてウィンドウを開き、お礼のメッセージを書いたHTMLファイルをロードしています。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function wopen(){ window.open("bay.html","WindowOpen1",
                  "toolbar=no,location=no,directories=
no,width=300,height=250");
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF" onUnload="wopen()">
*ページを抜ける時に新しいウィンドウを開く
</BODY>
</HTML>
```

注意

「onUnload」を使用する時の注意

イベントハンドラ「onUnload」を使って複雑な処理をさせると、OSやバージョンに関係なく、ブラウザの動作が不安定になる場合があります。
「onUnload」を使用する時は、十分なテストの上で使用してください。

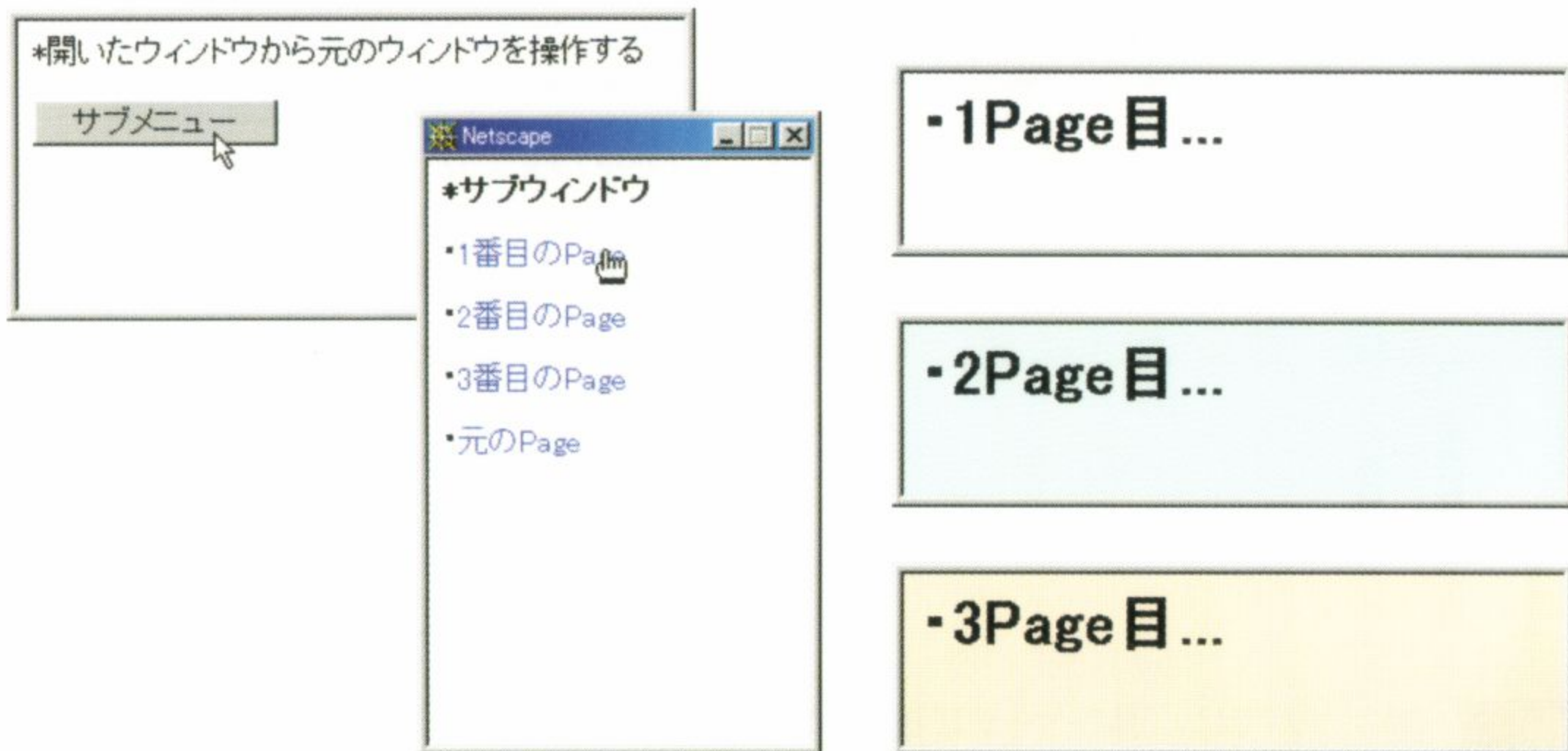
JavaScript 開いたウィンドウから元のウィンドウを操作する

opener [プロパティ]
closed [プロパティ]

*開いたウィンドウから元のウィンドウを操作する
サブメニュー
Netscape
*サブウィンドウ
・1番目のPage
・2番目のPage
・3番目のPage
・元のPage

・1Page目...

・2Page目...

・3Page目...

解説

「opener」は、元のウィンドウを参照するプロパティです。
サンプルでは、サブウィンドウ内のリンクがクリックされると、「opener.location.href」でサブウィンドウを開いたメインウィンドウが参照され、メインウィンドウに「GoWin()」内で指定しているURLがロードされます。
また、リンクがクリックされた時にメインウィンドウが閉じられていても、「closed」プロパティがウィンドウが閉じられている状態を取得しますので、「opener.closed」が真となって新たにウィンドウが開き、そこにページが読み込まれます。
JavaScript1.1で追加されたメソッドです。
Internet Explorerでは、「closed」プロパティがうまく機能しません。

Sample

【メインウィンドウ】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
function SWwopen(){
 window.open("menu.html","SWindow", "toolbar=no,location
=no,directories=no,status=no,menubar=no,scrollbars=no,
resizable=no,width=200,height=300");   }
//--->
</SCRIPT>
</HEAD>
```

```
<BODY BGCOLOR="#FFFFFF">
*開いたウィンドウから元のウィンドウを操作する<P>
<FORM>
 <INPUT TYPE="button" NAME= "Type9" VALUE=" サブメニュー " on
Click="SWwopen()">
</FORM>
</BODY>
</HTML>
```

【サブウィンドウ】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
<!--
function GoWin(WO) {
    if (opener.closed) {
        NewWin=window.open("","MWindow");
        NewWin.location.href=WO;
                              }
    else { opener.location.href=WO }
}
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<B>*サブウィンドウ</B>
<P>
·<A href="javascript:GoWin('page1.html')">1番目のPage</A>
<P>
·<A href="javascript:GoWin('page2.html')">2番目のPage</A>
<P>
·<A href="javascript:GoWin('page3.html')">3番目のPage</A>
<P>
·<A href="javascript:GoWin('14win.html')">元のPage</A>
<P>
</BODY>
</HTML>
```

JavaScript

NN4.0
NN4.06

JavaScript1.2で追加された window.open()の属性を使用する

window.open("*URL*","*ウィンドウ名*","*属性*") 【メソッド】

*JavaScript1.2で追加されたwindow.open()の属性を使用する

Open!!

200ピクセル

Win4.0 - Netscape

戻る 次 再読み込み ホーム 検索

•ウィンドウの外周:400x400(ピクセル)
•ディスプレイの左上角からの表示位置:200X200(ピクセル)

200ピクセル

解説

JavaScript1.2の「open()」メソッドで追加されたウィンドウ属性は、次の通りです。「outerWidth」と「outerHeight」は、それぞれウィンドウの外側の幅と高さを、ピクセル単位で指定します。外側のサイズなので、ツールバーやロケーションボックスなどがあってもなくても、ウィンドウのサイズは変わりません。「left」と「top」は、新しく開くウィンドウの左上角の位置を、ディスプレイの左上角の上からと左からのピクセル単位で指定します。

サンプルでは、ボタンをクリックしたタイミングで、ディスプレイの左上角から左へ200ピクセル・下へ200ピクセルの位置に左上角がくるように、ツールバーも含めて幅400ピクセル・高さ400ピクセルのウィンドウを開いています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function WO4_0(){
	var WO;
	WO=window.open("","Win4","toolbar=yes,location=no,dire
ctories=no,outerWidth=400,outerHeight=400,left=200,top=200");
```

```
        WO.document.open();
        WO.document.write("<HTML><TITLE>Win4.0</TITLE>          ");
        WO.document.write("</HEAD>                                ");
        WO.document.write("<BODY>                                 ");
        WO.document.write("・ウィンドウの外周:400x400(ピクセル)");
        WO.document.write("<BR>                                   ");
        WO.document.write("・ディスプレイの左上角からの表¥示¥位置:200X200(
ピクセル)");
        WO.document.write("</BODY>                                ");
        WO.document.write("</HTML>                                ");
        WO.document.close();
                }
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*JavaScript1.2で追加されたwindow.open()の属性を使用する<P>
<FORM NAME= "WO4">
 <INPUT TYPE="button" NAME= "wo4" VALUE="Open!!" onClick
="WO4_0()">
</FORM>
</BODY>
</HTML>
```

注意

「open()」の属性を使用する時の注意

JavaScript1.2で追加された「open()」の属性の内、「alwaysLowered」・「always Raised」・「z-lock」は、「Signed Script」内で設定する必要があります。

また、「innerWidth」・「innerHeight」・「outerWidth」・「outerHeight」でウィンドウサイズを100x100ピクセル以下に指定する場合や、「screenX」・「screenY」でディスプレイの表示領域外にウィンドウ表示位置を指定する時、「titlebar」の指定でタイトルバーを表示しないように指定する時にも、「Signed Script」内で設定する必要があります。

「alwaysLowered」・「alwaysRaised」・「z-lock」は、それを表示するプラットホームによって動きが異なる場合があります。例えば「z-lock」の場合、Windows95版では、他のアプリケーションのウィンドウが開いていても、それらより後ろに新しいウィンドウが開きます。ところが、Macintosh版の場合、他のアプリケーションのウィンドウがあると、そのウィンドウより前に新しいウィンドウが開く場合があります。

JavaScript

ウィンドウを前に出す

window.focus() 【メソッド】

Before

*ウィンドウを前に出す

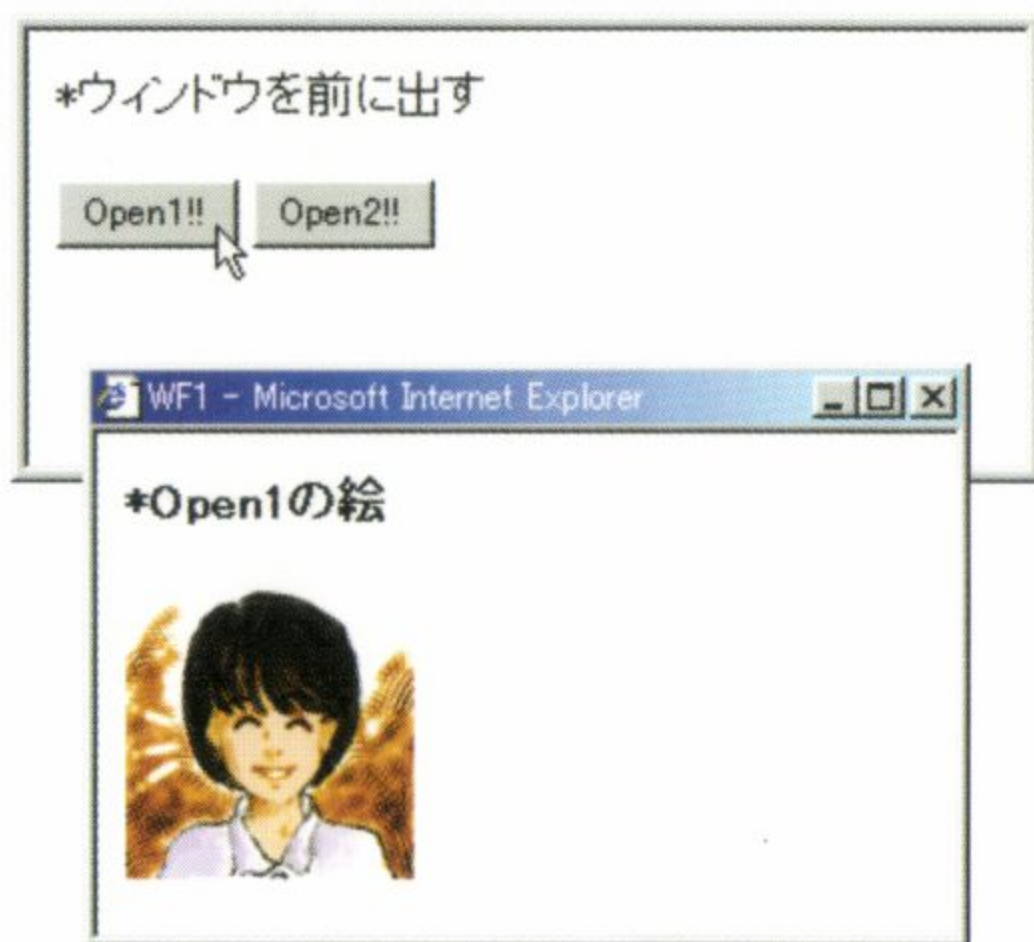
Open1!! Open2!!

After

*ウィンドウを前に出す

Open1!! Open2!!

WF1 - Microsoft Internet Explorer

*Open1の絵

解説

「focus()」は、ウィンドウにフォーカスをあたえるメソッドです。
サンプルでは、ウィンドウが新しく開いたり、新しく開いたウィンドウが書き変わった時に、もしもそのウィンドウが他のウィンドウの後ろに隠れていても、1番手前に移動します。
JavaScript1.1で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
function wopen1(){
    var WO1;
    WO1=window.open("","WindowFocus1", "toolbar=no,loca
tion=no,directories=no,width=300,height=450");
    WO1.focus();
    WO1.document.write("<HTML><TITLE>WF1</TITLE></HEAD>");
    WO1.document.write("<BODY BGCOLOR='#FFFFFF'>          ");
    WO1.document.write("<B>*Open1の絵</B>                ");
    WO1.document.write("<P>                               ");
    WO1.document.write("<IMG NAME='image1' SRC='image1.gif'
ALT='IMAGE1'> ");
    WO1.document.write("</BODY>                           ");
    WO1.document.write("</HTML>                           ");
    WO1.document.close();
        }
```

```
function wopen2(){
    var WO1;
    WO1=window.open("","WindowFocus1", "toolbar=no,loca
tion=no,directories=no,width=300,height=450");
    WO1.focus();
    WO1.document.write("<HTML><TITLE>WF2</TITLE></HEAD>");
    WO1.document.write("<BODY BGCOLOR='#FFFFFF'>           ");
    WO1.document.write("<B>*Open2の絵</B>          ");
    WO1.document.write("<P>                              ");
    WO1.document.write("<IMG NAME='image2' SRC='image2.
gif' ALT='IMAGE2'> ");
    WO1.document.write("</BODY>                          ");
    WO1.document.write("</HTML>                          ");
    WO1.document.close();
        }
//---->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ウィンドウを前に出す<P>
<FORM>
 <INPUT TYPE="button" NAME= "Type11" VALUE="Open1!!" on
Click="wopen1()">
 <INPUT TYPE="button" NAME= "Type11" VALUE="Open2!!" on
Click="wopen2()">
</FORM>
</BODY>
</HTML>
```


注意 小さなウィンドウを開くときの注意

「window.open()」のウィンドウサイズ指定で、ウィンドウの横幅がステータスバー内のファイルの読み込み状況を示すバーよりも小さく指定されていると、Macintosh PPC版Netscape Navigator3.0では、システムエラーを起こすことがあります。

JavaScript

ウィンドウを後ろに送る

window.blur() [メソッド]

Before

*ウィンドウを後ろに送る

始めにウインドウを開きます

*下のボタンを押すと、このウインドウが後ろに移動します。

後ろへ!!

After

*ウィンドウを後ろに送る

始めにウインドウを開きます

*下のボタンを押すと、

後ろへ!!

解説

「blur()」メソッドは、ウィンドウからフォーカスを移動させます。
サンプルでは、複数のウィンドウが開いている時にボタンが押されると「window.blur()」が評価され、ウィンドウが後ろに回ります。
JavaScript1.1で追加されたメソッドです。

Sample

```
<FORM>
 <INPUT TYPE="button" NAME= "WindowBlur1" VALUE=" 後ろへ!! "
 onClick="window.blur()">
</FORM>
```

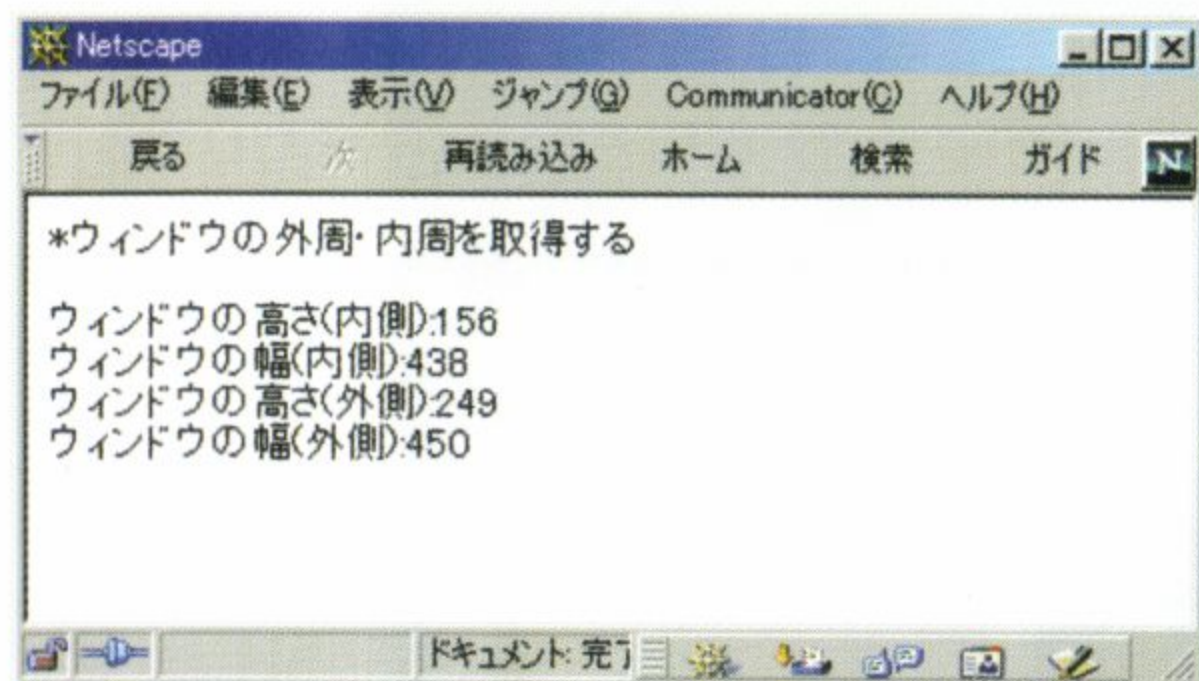
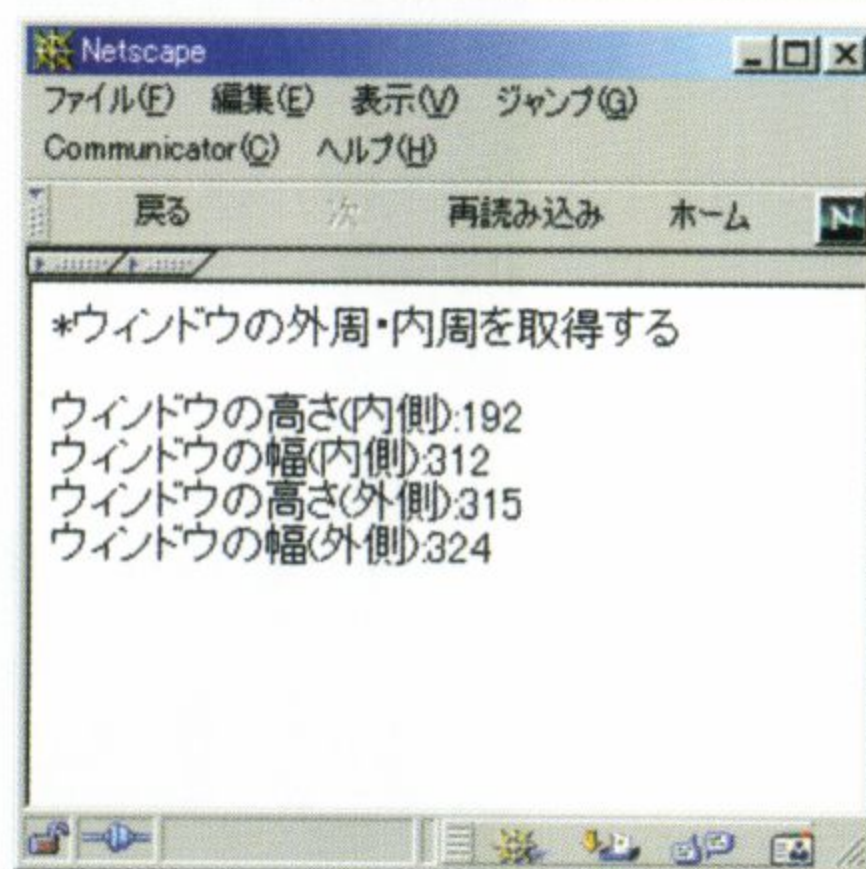
ウィンドウの外周・内周を取得する

window.innerHeight 【プロパティ】
window.innerWidth 【プロパティ】
window.outerHeight 【プロパティ】
window.outerWidth 【プロパティ】

Before

Netscape
ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)
戻る 次 再読み込み ホーム 検索 ガイド
*ウィンドウの外周・内周を取得する
ウィンドウの高さ(内側):156
ウィンドウの幅(内側):438
ウィンドウの高さ(外側):249
ウィンドウの幅(外側):450
ドキュメント: 完了

After

Netscape
ファイル(F) 編集(E) 表示(V) ジャンプ(G)
Communicator(C) ヘルプ(H)
戻る 次 再読み込み ホーム
*ウィンドウの外周・内周を取得する
ウィンドウの高さ(内側):192
ウィンドウの幅(内側):312
ウィンドウの高さ(外側):315
ウィンドウの幅(外側):324

解説

「innerHeight」・「innerWidth」プロパティはウィンドウ内の表示領域の高さと幅の値を、「outerHeight」・「outerWidth」プロパティはツールバーやステータスバーなども含めたウィンドウの外側の高さと幅の値を、それぞれ持っています。
これらは、JavaScript1.2で追加されたプロパティです。

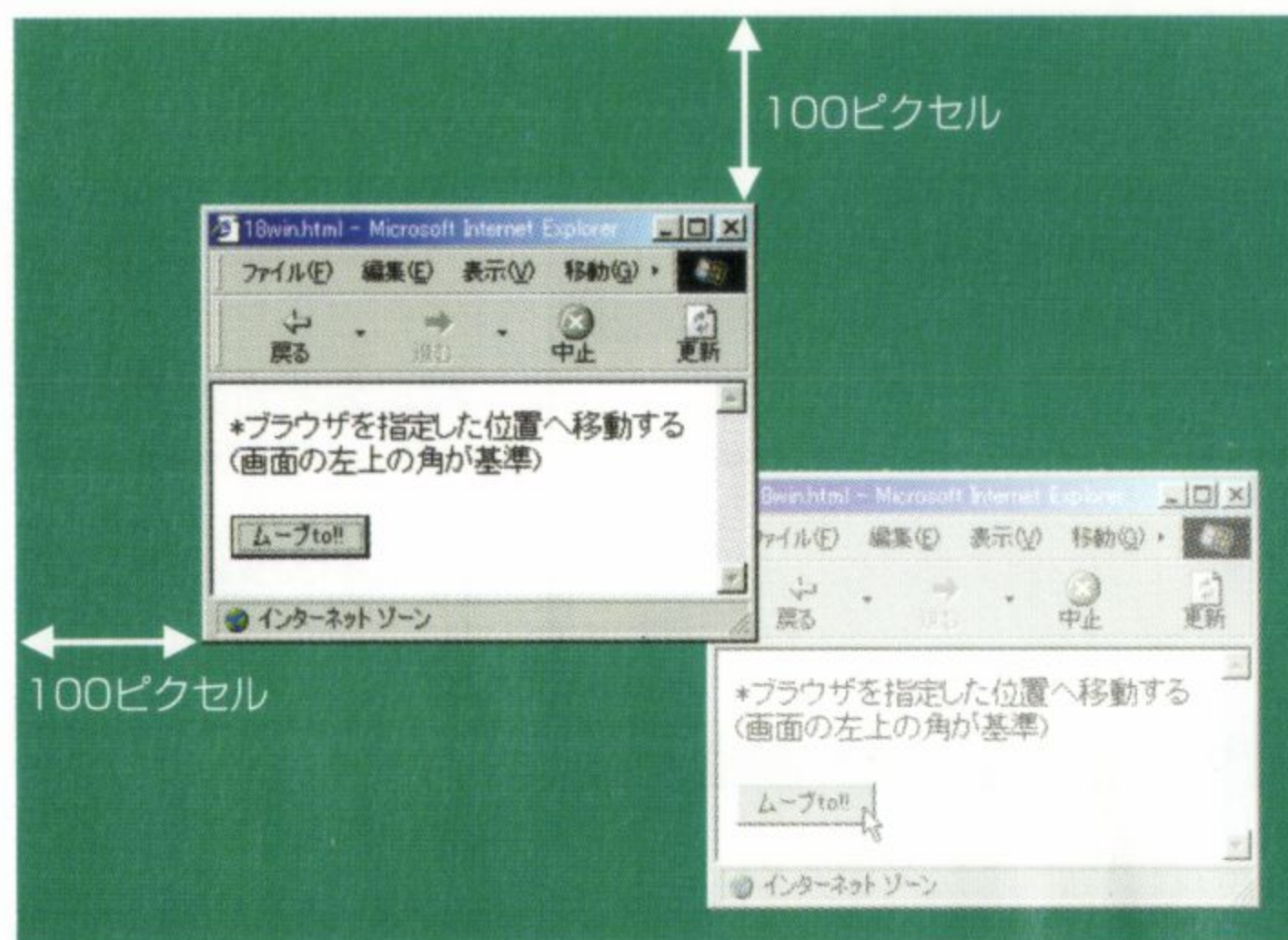
Sample

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*ウィンドウの外周・内周を取得する<P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
    document.write("ウィンドウの高さ(内側):",window.innerHeight);
    document.write("<BR>");
    document.write("ウィンドウの幅(内側):",window.innerWidth);
    document.write("<BR>");
    document.write("ウィンドウの高さ(外側):",window.outerHeight);
    document.write("<BR>");
    document.write("ウィンドウの幅(外側):",window.outerWidth);
//-->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

ブラウザを指定した位置へ移動する

window.moveTo(*x*,*y*) [メソッド]

x	ディスプレイ左上から下方向（ピクセル）
y	ディスプレイ左上から右方向（ピクセル）

解説

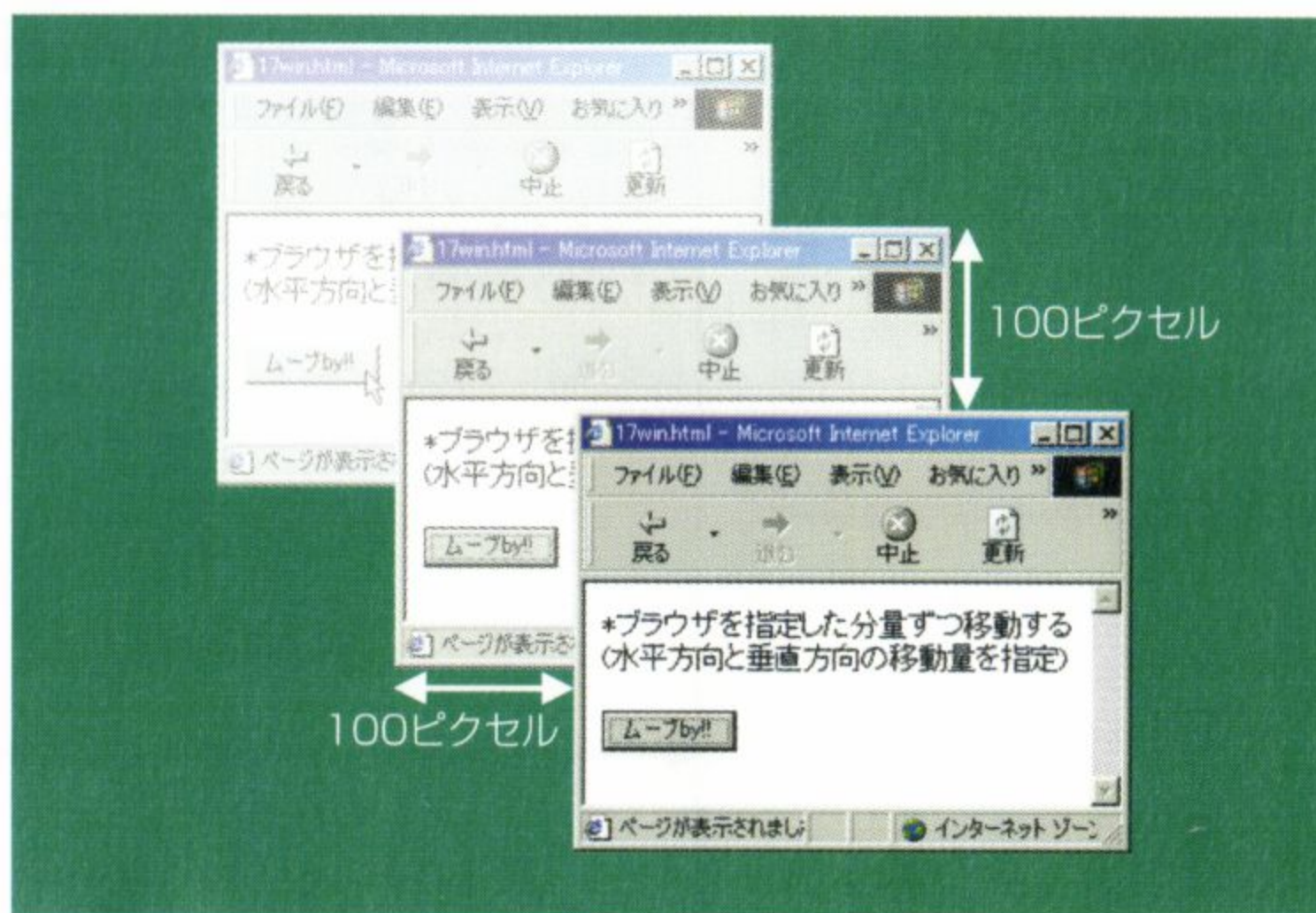
windowオブジェクトの「moveTo()」メソッドは、ピクセル単位で指定した位置(x,y)へ、ウィンドウを移動させます。
サンプルでは、ウィンドウがどこの位置に表示されていても、ボタンをクリックすることによって、ディスプレイの左上角から下へ100ピクセル・右へ100ピクセルの位置にウィンドウの左上角がくるように移動します。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function MVto(){ window.moveTo(100,100) }
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ブラウザを指定した位置へ移動する(画面の左上の角が基準)<P>
<FORM NAME="MVTO">
 <INPUT TYPE="button" NAME="mvto" VALUE="ムーブ to!!" on
Click="MVto()">
</FORM>
</BODY>
</HTML>
```

ブラウザを指定した分量ずつ移動する

window.moveBy(*x*,*y*) 【メソッド】

x	水平方向への移動量（ピクセル）
y	垂直方向への移動量（ピクセル）

解説

「moveBy()」メソッドは、ピクセル単位でウィンドウを移動させます。
サンプルでは、ボタンをクリックするごとに、ウィンドウが右へ100ピクセル・下へ100ピクセルずつ移動します。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function MVby(){ window.moveBy(100,100) }
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ブラウザを指定した分量ずつ移動する(水平方向と垂直方向の移動量を指定)<P>
<FORM NAME="MVBY">
 <INPUT TYPE="button" NAME="mvby" VALUE="ムーブ by!!" on
Click="MVby()">
</FORM>
</BODY>
</HTML>
```

JavaScript

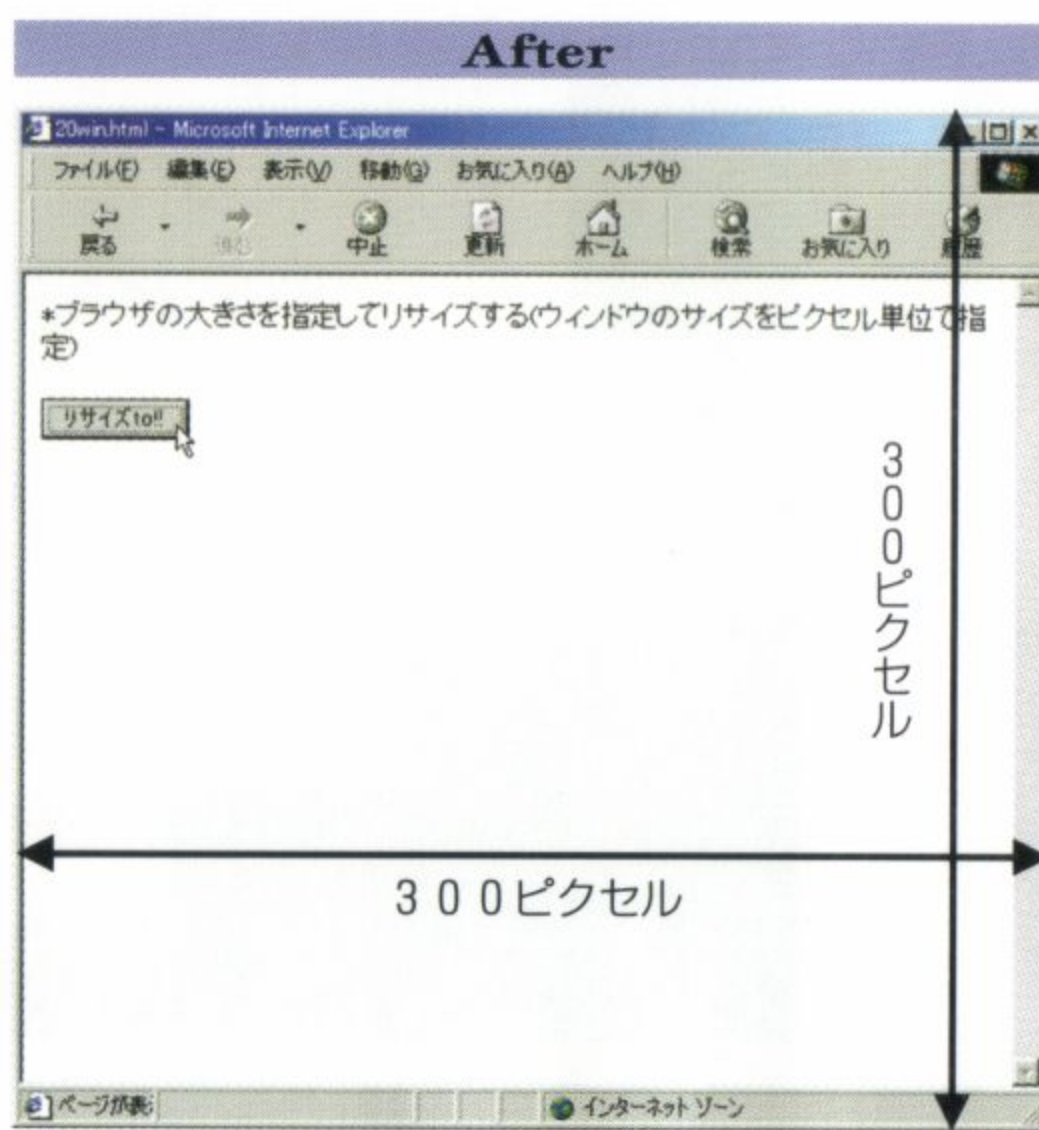
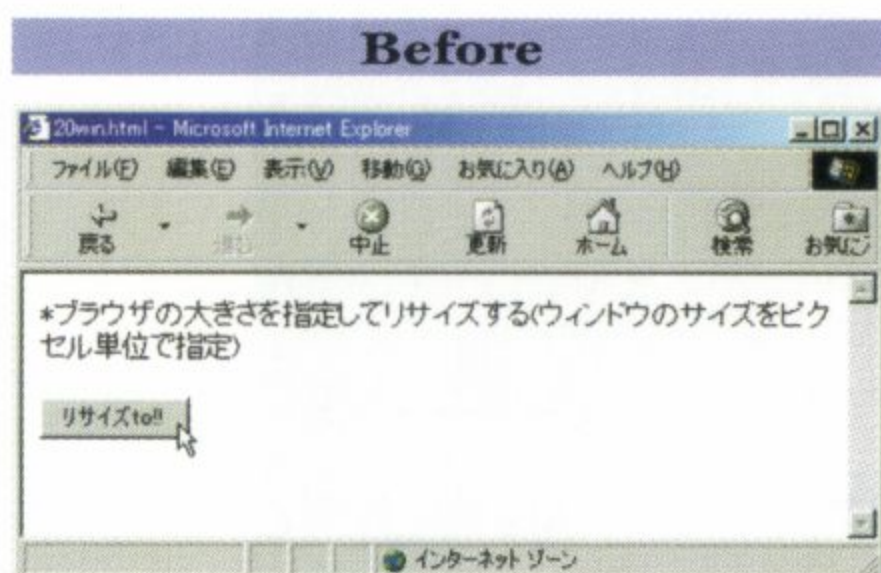
ブラウザの大きさを指定してリサイズする

window.resizeTo(x,y) [メソッド]

x ウィンドウ左上角から垂直方向（ピクセル）

y ウィンドウ左上角から水平方向（ピクセル）

Before

After

300ピクセル

300ピクセル

解説

「resizeTo()」メソッドは、ピクセル単位で指定したサイズにウィンドウをリサイズします。

サンプルでは、ボタンをクリックすると、ウィンドウの左上角を基準に、縦横300ピクセルにリサイズします。

JavaScript1.2で追加されたメソッドです。

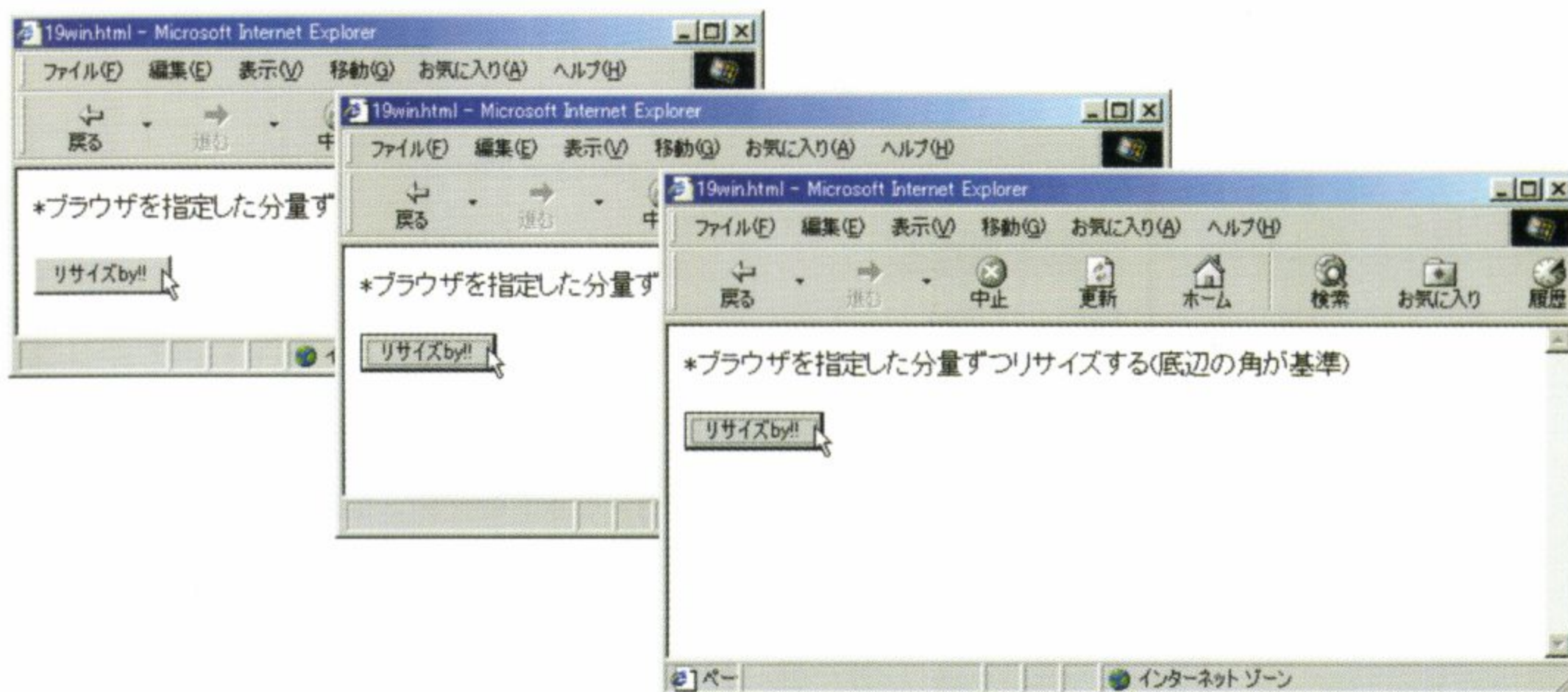
Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function RSto(){ window.resizeTo(300,300) }
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ブラウザの大きさを指定してリサイズする(ウィンドウのサイズをピクセル単位で指定)<P>
<FORM NAME="RESTO">
 <INPUT TYPE="button" NAME="resto" VALUE="リサイズto!!" on
Click="RSto()">
</FORM>
</BODY></HTML>
```

ブラウザを指定した分量ずつリサイズする

window.resizeBy(x,y) [メソッド]

x	底辺のリサイズ量(ピクセル)
y	右辺のリサイズ量(ピクセル)

19win.html - Microsoft Internet Explorer
*ブラウザを指定した分量ずつリサイズする(底辺の角が基準)
リサイズby!!

解説

「resizeBy()」メソッドは、ウィンドウのサイズをピクセル単位で変更します。
サンプルでは、ウィンドウの底辺と右辺が、ボタンをクリックするごとにそれぞれ50ピクセルずつ広がります。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function RSby(){ window.resizeBy(50,50) }
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ブラウザを指定した分量ずつリサイズする(底辺の角が基準)<P>
<FORM NAME="RESBY">
<INPUT TYPE="button" NAME="resby" VALUE="リサイズ by!!" on
Click="RSby()">
</FORM>
</BODY>
</HTML>
```

ウィンドウをスクロールする

window.scroll(*x*,*y*) [メソッド]

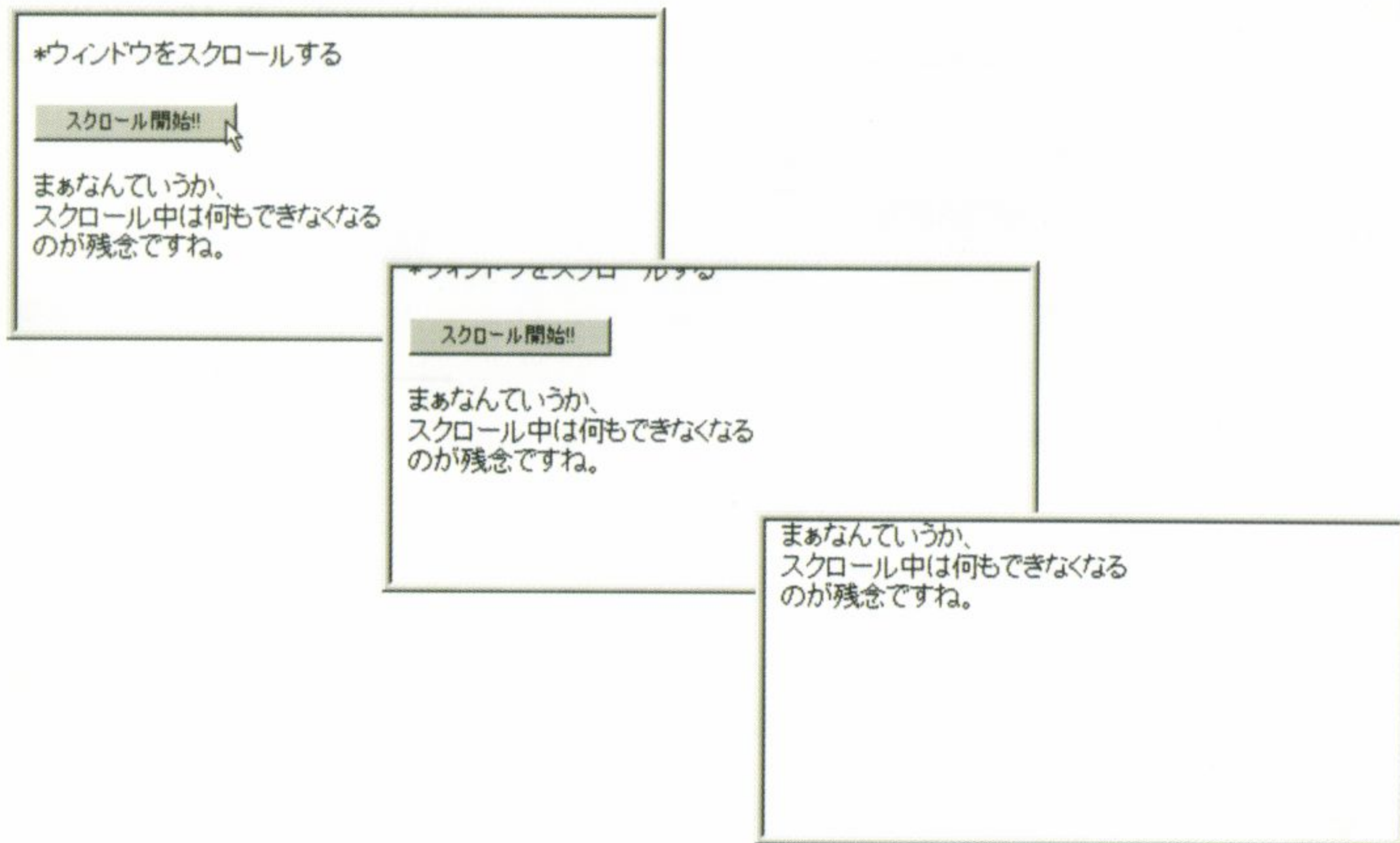
*ウィンドウをスクロールする

スクロール開始!!

まぁなんていうか、
スクロール中は何もできなくなる
のが残念ですね。

スクロール開始!!

まぁなんていうか、
スクロール中は何もできなくなる
のが残念ですね。

まぁなんていうか、
スクロール中は何もできなくなる
のが残念ですね。

解説

「scroll()」メソッドは、x軸・y軸をピクセルで指定し、指定した位置までブラウザをスクロールさせます。
サンプルでは、for文でy軸に数値を代入し続け、ブラウザを縦方向にスクロールさせています。
for文で式をループさせているだけなので、実行速度はマシンパワーに左右され、実行中は他の動作を受け付けなくなります。
JavaScript1.1で追加されたメソッドですが、Netscape Navigator2.0でも動作します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function SCLL() {
    for(var i=0; i<150;i++) { window.scroll(0,i) }
    for(i=149; i>=0;i--) { window.scroll(0,i) }
}
//--->
</SCRIPT>
</HEAD>
```

```
<BODY BGCOLOR="#FFFFFF">
*ウィンドウをスクロールする<P>
<FORM>
 <INPUT TYPE="button" NAME= "SCLL1" VALUE="スクロール開始!!" 
onClick="SCLL()">
</FORM>
<P>
まぁなんていうか、<BR>
スクロール中は何もできなくなる<BR>
のが残念ですね。
<BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>
ダミーです...。
</BODY>
</HTML>
```


注意

「scroll()」を使用する時の注意

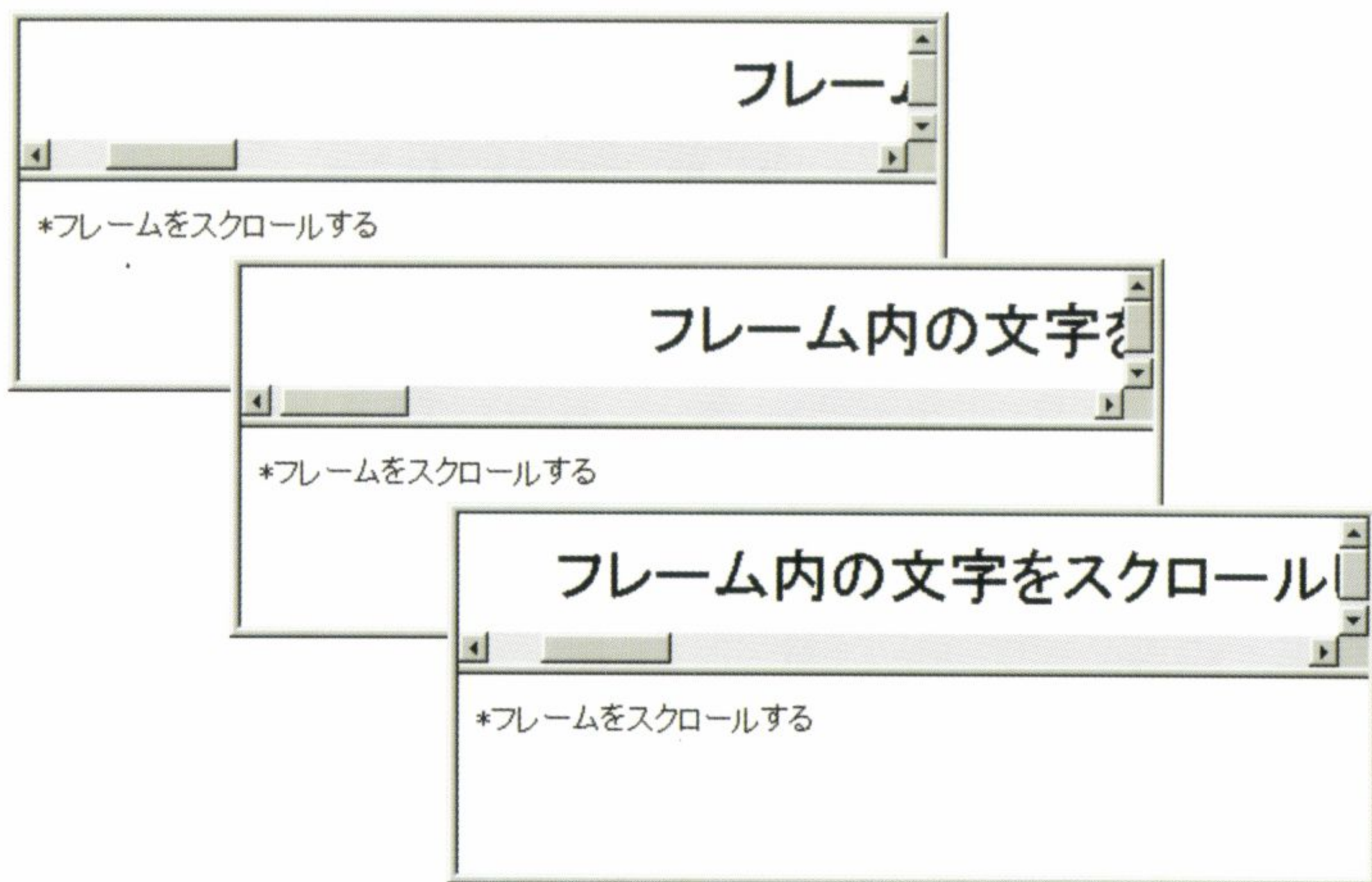
Windows版のNetscape Navigatorでは、"scroll(x,y)"でスクロール位置を指定していても、その範囲にドキュメントがなければドキュメントがある範囲しかスクロールしません。

また、フレームや「window.open()」の指定でスクロールバーを出さないように指定していてもスクロールしないので、注意してください。

JavaScript

フレームをスクロールする

window.scroll(*x*,*y*) 【メソッド】

x	ウィンドウ左上角から垂直方向（ピクセル）
y	ウィンドウ左上角から水平方向（ピクセル）

解説

サンプルでは、フレーム内のウィンドウを、「scroll()」メソッドのx軸の値に「setTimeout()」で、一定時間ごとにiの値に1ずつ加えながら代入することによって、横にスクロールさせています。
一定時間に処理を繰り返しているので、実行中は他の動作を受け付けなくなるようなことはありません。
スクロールさせる文字は、<NOBR>タグを使って改行されないようにしています。
JavaScript1.1で追加されたメソッドですが、Netscape Navigator2.0でも動作します。

Sample

【フレーム部分】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<FRAMESET ROWS="80,*">
    <FRAME SRC="f1.html" name="f1">
    <FRAME SRC="f2.html" name="f2">
</FRAMESET>
```

```
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
var i = 0 ;
var MOZI = 3200 ;
var ST = 50 ;
function SCLL2() {
    if ( i < MOZI) { i = i+2 ;
           window.scroll(i,0) ;
           }
             setTimeout("SCLL2()", ST) ;
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF" onLoad="SCLL2()">
<NOBR>
<FONT SIZE=6>                                          フレー
ム内の文字をスクロールします...。NOBRの指定とsetTimeoutがポイントです。for
文を使うとスクロール中は何もできなくなってしまいます。Windowsではスクロールバー
を"No"にしているとうまく行かない時があります。</FONT>
</NOBR>
</BODY>
</HTML>
```

【f2.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*フレームをスクロールする
</BODY>
</HTML>
```

JavaScript

ブラウザを指定した位置までスクロールする

window.scrollTo(*x*,*y*) [メソッド]

x	ウィンドウ左上角から垂直方向（ピクセル）
y	ウィンドウ左上角から水平方向（ピクセル）

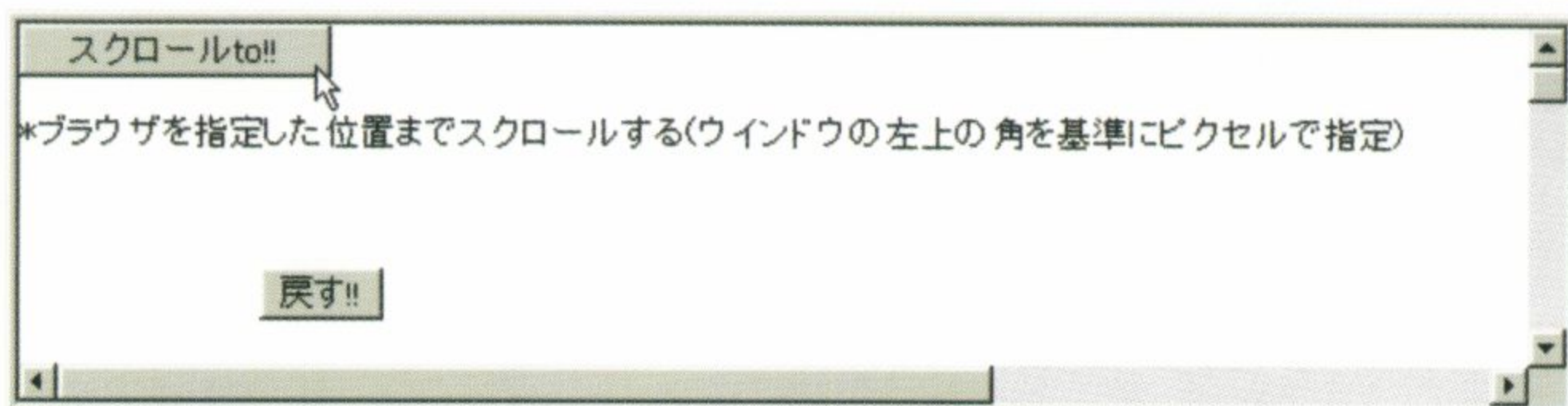
Before

スクロールto!!

*ブラウザを指定した位置までスクロールする(ウインドウの左上の角を基準にピクセルで指定)

戻す!!

After

戻す!!

解説

「scrollTo()」メソッドは、JavaScript1.1の「scroll()」メソッドを拡張したもので、ピクセル単位で指定した位置へ、ウィンドウの表示領域を移動させます。
「scroll()」メソッドも、以前までと互換を保つため使用可能になっています。
サンプルでは、「スクロールto!!」ボタンをクリックすると、ウィンドウの表示領域の左上角から、上へ100ピクセル・左へ100ピクセルの位置にウィンドウの表示領域が移動し、「戻す!!」ボタンを押すと元の位置へ戻ります。scrollBy()(次項)と違い、1度表示領域が移動すると、それ以降は、ボタンをクリックしても表示位置は変わりません。
表示領域はスクロールバーが出ている範囲でしか移動できないので、サンプルでは、1000×1000ピクセルのレイヤーを設定することによって、スクロールバーが出るようにしています。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function SRto1(){ window.scrollTo(100,100) }
function SRto2(){ window.scrollTo(-100,-100) }
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<LAYER NAME="LAY1" TOP=0 LEFT=0 WIDTH=1000 HEIGHT=1000>
<FORM>
 <INPUT TYPE="button" NAME="srto" VALUE="スクロールto!!" on
Click="SRto1()">
</FORM>
*ブラウザを指定した位置までスクロールする(ウインドウの左上の角を基準にピクセルで指定)
</LAYER>
<LAYER NAME="LAY2" TOP=100 LEFT=100 WIDTH=30 HEIGHT=1000>
 <FORM>
        <INPUT TYPE="button" NAME="srto" VALUE="戻す!!" on
Click="SRto2()">
 </FORM>
</LAYER>
</BODY>
</HTML>
```

NN4.0
NN4.06

JavaScript

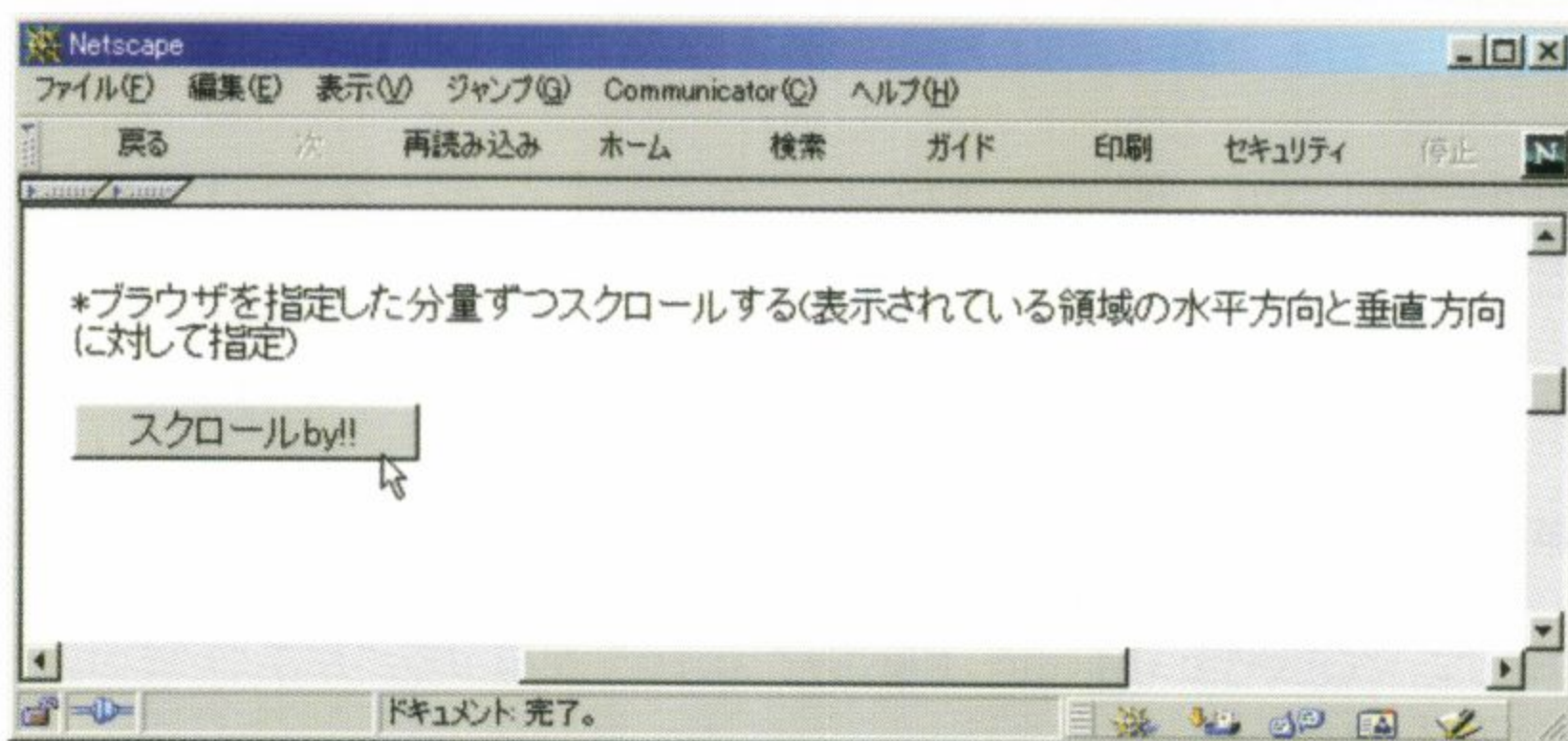
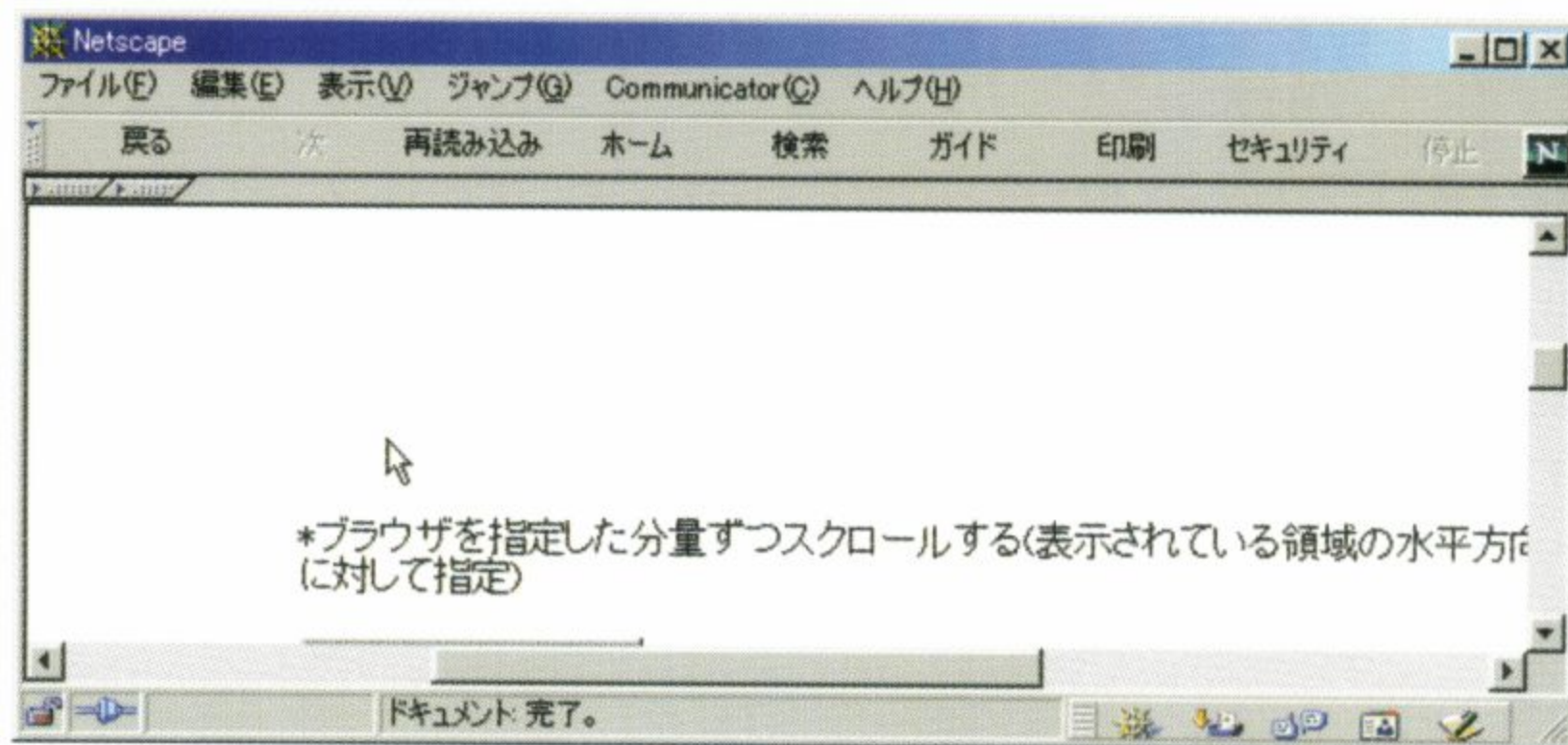
ブラウザを指定した分量ずつスクロールする

```
window.scrollBy(x,y)                                   [メソッド]
    x                                       水平方向（ピクセル）
    y                                       垂直方向（ピクセル）
```

Before

Netscape
ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)
戻る 次 再読み込み ホーム 検索 ガイド 印刷 セキュリティ 停止
*ブラウザを指定した分量ずつスクロールする(表示されている領域の水平方向と垂直方向に対して指定)
スクロールby!!
ドキュメント: 完了。

After

Netscape
ファイル(F) 編集(E) 表示(V) ジャンプ(G) Communicator(C) ヘルプ(H)
戻る 次 再読み込み ホーム 検索 ガイド 印刷 セキュリティ 停止
*ブラウザを指定した分量ずつスクロールする(表示されている領域の水平方向に対して指定)
ドキュメント: 完了。

解説

「scrollBy()」メソッドは、ウィンドウの表示領域をピクセル単位で移動します。サンプルでは、ボタンをクリックするごとに、表示領域が右へ100ピクセル・下へ100ピクセルずつ移動します。
Windows版のNetscape Navigatorの場合、表示領域はスクロールバーが出ている範囲でしか移動できません。そこでサンプルでは、1500×1500ピクセルのレイヤーを設定することによってスクロールバーが出るようにし、その親レイヤー内の上から500ピクセル・左端から500ピクセルの位置に子レイヤーを作り、そこにボタンを設定しています。なお、Macintosh版のNetscape Navigatorは、スクロールバーが出ていない場合でもスクロール可能です。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function SRby(){ window.scrollBy(-100,-100) }
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<LAYER NAME="LAY1" TOP=0 LEFT=0 WIDTH=1500 HEIGHT=1500>
    <LAYER NAME="LAY1" TOP=500 LEFT=500 WIDTH=600 HEIGHT
=100>
*ブラウザを指定した分量ずつスクロールする(表示されている領域の水平方向と垂直方向に
対して指定)
<FORM>
 <INPUT TYPE="button" NAME="srby" VALUE="スクロールby!!" on
Click="SRby()">
</FORM>
    </LAYER>
</LAYER>
</BODY>
</HTML>
```

ウィンドウ内の文字を検索する

window.find(文字列,[true | false]) [メソッド]

Before

*ウィンドウ内の文字を検索する

"JavaScript"とは、Netscape社がウエッブページの処理能力を高めるために開発した"LiveScript"を元に、Netscape社とSunが共同で開発したScript言語で、現在Netscape Navigator 2.0以降のブラウザとInternet Explorer 3.0以降のブラウザで対応されています。

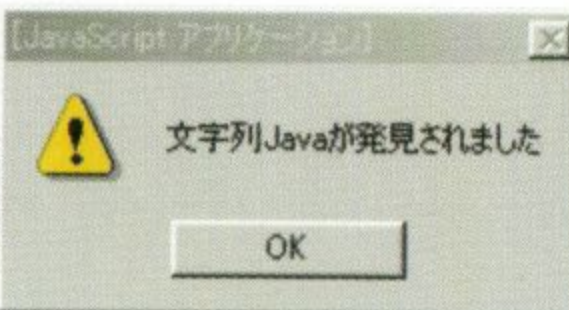
Java 検索!!

After

*ウィンドウ内の文字を検索する

[JavaScript アプリケーション]

文字列Javaが発見されました

OK

Java 検索!!

解説

「find()」メソッドは、ウィンドウ内の文字列を検索し指定された文字列が発見された時に、true(真)またはfalse(偽)を返すように設定することができます。
サンプルでは、フォームに入力された文字列がウィンドウ上に含まれているかどうかを検索し、結果を警告用のダイアログボックスに表示しています。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function FIN(i) {
  if ( window.find( i,true) ) { alert ("文字列" + i + "が発見
されました") }
```

```
 else { alert ("文字列" + i + "は発見されませんでした") }
}
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ウィンドウ内の文字を検索する<P>
<HR>
<BLOCKQUOTE>
<B>"JavaScript"</B>とは、Netscape社がウエップページの処理能力を高める
ために開発した<B>"LiveScript"</B>を元に、Netscape社とSunが共同で開発
した<B>Script言語</B>で、現在Netscape Navigator 2.0以降のブラウザ
とInternet Explorer 3.0以降のブラウザで対応されています。
</BLOCKQUOTE>
<HR>
<FORM>
 <INPUT TYPE="text" NAME="fin1" VALUE="" SIZE=30>
 <INPUT TYPE="button" NAME= "fin2" VALUE="検索!!" onClick
="FIN(fin1.value)">
</FORM>
</BODY>
</HTML>
```

JavaScript

NN4.0
NN4.06

ブラウザを制御するボタンを作る

opener.back()	←「戻る」ボタン	[メソッド]
opener.forward()	←「進む」ボタン	[メソッド]
opener.home()	←「ホーム」ボタン	[メソッド]
opener.stop()	←「停止」ボタン	[メソッド]

*ブラウザを制御するボタンを作る

Open!!

Back!! Forward!! Home!! Stop!!

解説

「back()」メソッドはブラウザの「戻る」(Back)ボタンで参照されるのと同じ1つ前のウィンドウのURLを、「forward()」メソッドはブラウザの「進む」(Forward)ボタンで参照されるのと同じ1つ先のウィンドウのURLを、「home()」メソッドはブラウザの「ホーム」(Home)ボタンで参照されるのと同じURLをそれぞれ返します。「stop()」メソッドは、ブラウザの「停止」(Stop)ボタンを押した時と同じ状態を返します。

サンプルでは、ブラウザのメニューバーのボタンと同じ働きを持ったボタンがあるサブウィンドウを開き、そこから元のウィンドウを操作できるようにしています。

また、「open()」メソッドの属性内の、「innerWidth」・「innerHeight」は、ウィンドウの表示領域のサイズをピクセル単位で指定します。

JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function NewWin(){
 var NW;
 NW=window.open("","NewWin1","toolbar=no,location=no,di
rectories=no,innerWidth=   400,innerHeight=100");
 NW.document.open();
 NW.document.write("<HTML><HEAD><TITLE>WIN4.0_SAMPLE</TI
TLE>");
 NW.document.write("</HEAD>");
 NW.document.write("<BODY>");
 NW.document.write("<FORM>");
 NW.document.write("<INPUT TYPE='button' NAME='back' VAL
UE='Back!!' onClick='opener.back()'>");
 NW.document.write("<INPUT TYPE='button' NAME='forward'
VALUE='Forward!!' onClick='opener.forward()'>");
```

```
 NW.document.write("<INPUT TYPE='button' NAME='home' VAL
UE='Home!!' onClick='opener.home()'>");
 NW.document.write("<INPUT TYPE='button' NAME='stop' VAL
UE='Stop!!' onClick='opener.stop()'>");
 NW.document.write("</FORM>");
 NW.document.write("</BODY>");
 NW.document.write("</HTML>");
 NW.document.close();
}
//-->
</SCRIPT>
</HEAD>
<BODY>
*ブラウザを制御するボタンを作る<P>
<FORM>
<INPUT TYPE="button" NAME="winopen" VALUE="Open!!" onCli
ck="NewWin()">
</FORM>
</BODY>
</HTML>
```

JavaScript

NN4.0

NN4.06

各種バーを制御する

window.locationbar.visible = *[true|false]|[1/0]* 【プロパティ】
window.menubar.visible = *[true|false]|[1/0]* 【プロパティ】
window.personalbar.visible = *[true|false]|[1/0]* 【プロパティ】
window.scrollbars.visible = *[true|false]|[1/0]* 【プロパティ】
window.statusbar.visible = *[true|false]|[1/0]* 【プロパティ】
window.toolbar.visible = *[true|false]|[1/0]* 【プロパティ】

解説

JavaScript1.2対応のNetscape Navigator4.0では、メニューバーやステータスバーなどの各種バーをプロパティとして持っています。
各種バーは、window.locationbar.visible=trueやwindow.locationbar.visible=1というように".visible=[true | 1]"と指定されている時は、表示状態になります。逆に、window.locationbar.visible=falseやwindow.locationbar.visible=0という様に「.visible=[false | 0]」と指定されている時は、非表示状態になります。
これらのプロパティの値は、後からでも変更可能です。
これらのプロパティは、「Signed Script」内で設定する必要があります。
JavaScript1.2で追加されたプロパティです。

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript

入力されたURLを別フレームに表示する

parent.フレーム名.プロパティ

Before

*入力されたURLを別フレームに表示する

http://www.shuwasystem.co.jp/ Go!!

After

Shuwa System

Contents
▼書籍情報

だれでもカンペキ
Windows98

http://www.shuwasystem.co.jp/ Go!!

解説

JavaScriptが記述しているフレーム以外のフレームにJavaScriptの値を引き渡す時は、「parent.値を渡すフレーム名.値」と指定します。
サンプルでは、フォーム内に入力されたURLの値を「parent.f1.location」として、フレームネーム「f1」の「ロケーションの値」に引き渡しています。
一見ややこしそうですが、慣れれば非常に簡単で利用価値の高い用法なので、ぜひ1度チャレンジしてみてください。
なお、「location」プロパティの使い方に関しては、「locationオブジェクト」の「入力されたURLへ進むフォームを作る」(P.354)を参照してください。
また、フレームを抜けてページを表示するには、「parent.top.location.href=URL」と「top」プロパティを指定します。具体的な使い方は、「Formオブジェクト」の「ラジオボタンをリンクに使う」(P.370)を参考にしてください。

Sample

【フレーム】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<FRAMESET ROWS="*,60">
    <FRAME SRC="f1.html" NAME="f1">
    <FRAME SRC="f2.html" NAME="f2">
</FRAMESET>
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*入力されたURLを別フレームに表示する
</BODY>
</HTML>
```

【f2.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function LC2(go){
    if (go.url2.value != "") { parent.f1.location.href=
go.url2.value }
    else { alert("URLを入力してください") }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<FORM NAME="URL2">
<INPUT TYPE="text" NAME="url2" VALUE="http://" SIZE=40 >
<INPUT TYPE="button" NAME= "CF2" VALUE=" Go!! " onClick=
"LC2(this.form)">
</FORM>
</BODY>
</HTML>
```

<FRAMESET>→「フレーム」の「フレームの全体構成を指定する」:P.168参照
<NOFRAMES>→「フレーム」の「フレームに未対応のブラウザ向けの内容を入れる:P.176参照
location→「locationオブジェクト」の「自ページのURLを取得する」:P.353参照

JavaScript

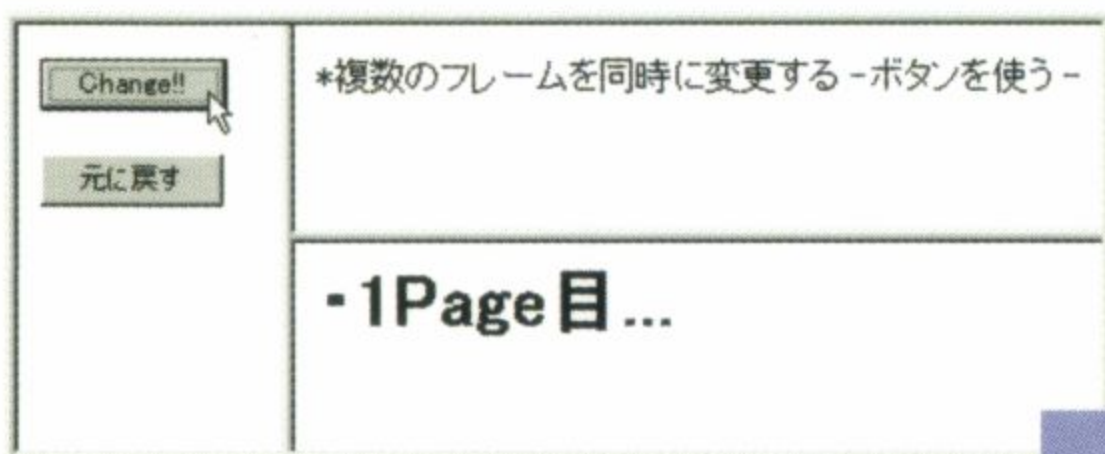
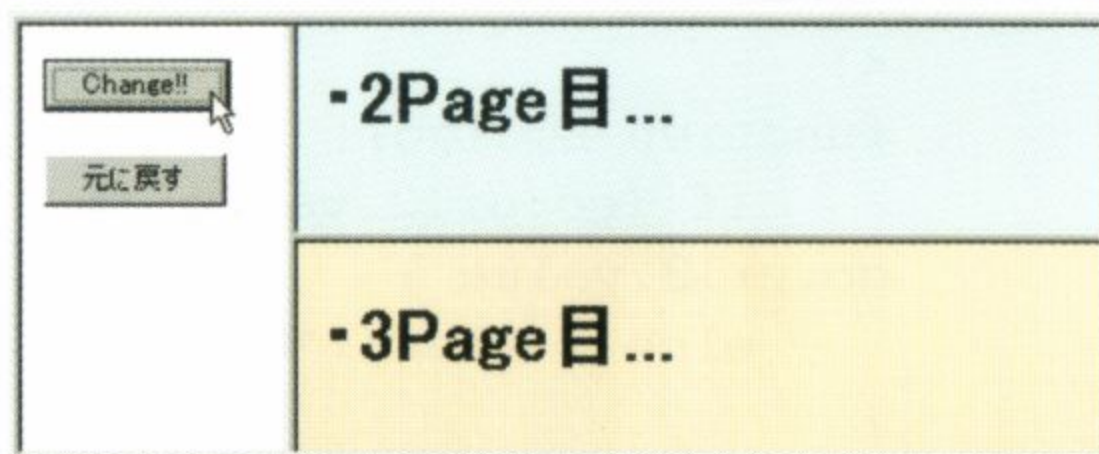
複数のフレームを同時に変更する - ボタンを使う -

parent.フレーム名.location.href=*URL*

Before

Change!!
元に戻す
*複数のフレームを同時に変更する - ボタンを使う -
・1Page目...

After

Change!!
元に戻す
・2Page目...
・3Page目...

解説

サンプルでは、ボタンがクリックされた時に、2つのURLの値を持った関数を発生し、その値を1度に関数の処理へ引き渡すことにより、複数のフレームに新しいページを読み込んでいます。このように、「parent.フレーム名.location.href=URL」の処理を複数設定するだけで、1回の処理で複数のフレームを変更することが可能です。実際に試す場合には、この他にも"page1.html"〜"page3.html"の3つのHTMLファイルを用意してください。

Sample

【フレーム】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<FRAMESET COLS="120,*">
    <FRAME SRC="f1.html" NAME="f1">
  <FRAMESET ROWS="50%,50%">
    <FRAME SRC="f2.html" NAME="f2">
    <FRAME SRC="page1.html" NAME="f3">
   </FRAMESET>
</FRAMESET>
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function CH1(P1,P2){
    parent.f2.location.href=P1;
    parent.f3.location.href=P2;
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<FORM NAME="CHANG1">
<INPUT TYPE="button" NAME= "chang1" VALUE=" Change!! " on
Click="CH1('page2.html','page3.html')">
</FORM>
<P>
<FORM name="CHANG2">
<INPUT TYPE="button" NAME= "chang2" VALUE=" 元に戻す " on
Click="CH1('f2.html','page1.html')">
</FORM>
</BODY>
</HTML>
```

【f2.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*複数のフレームを同時に変更する - ボタンを使う -<P>
</BODY>
</HTML>
```

window.open()→「windowオブジェクト」の「新しいウィンドウを開く」:P.296参照
<INPUT TYPE="button">→「Formオブジェクト」の「ボタンをリンクに使う」:P.372参照

注意

フレーム内にページをロードする時の注意

frameオブジェクトを使用して、「parent.フレーム名.location.href=URL」としてフレームに新しいページをロードしようとした時は、Macintosh版のNetscape Navigator2.0などの一部ブラウザでは、2つ目以降のURLが引けずにエラーになってしまう場合があります。

そのような場合は、「複数のフレームを同時に変更する - リンクを使う -」(次ページ)を参照してください。

JavaScript

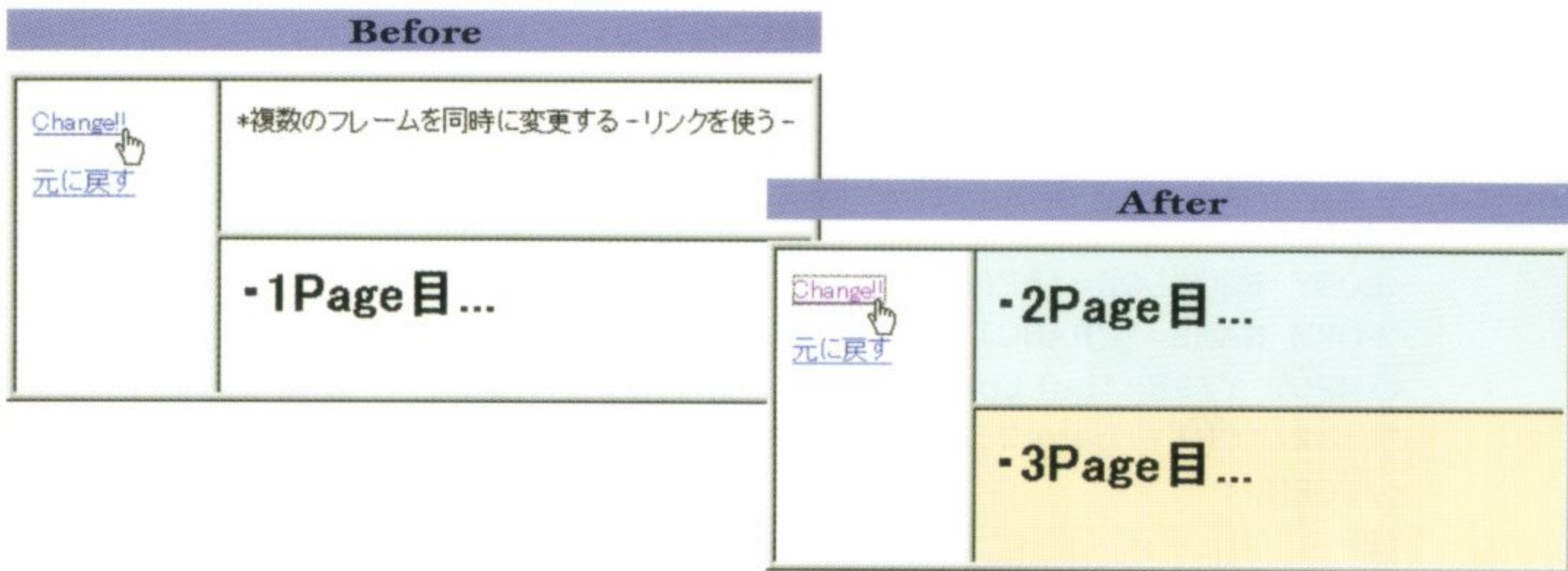
複数のフレームを同時に変更する -リンクを使う-

window.open(*URL*, *フレーム名*)
HREF="JavaScript:*関数*"

Before

Change!!
元に戻す
*複数のフレームを同時に変更する -リンクを使う -
・1Page目…

After

Change!!
元に戻す
・2Page目…
・3Page目…

解説

サンプルの「window.open(URL,フレーム名)」の部分は、frameオブジェクトというよりは、windowオブジェクト的な用法です。
サンプルでは、リンクがクリックされた時に2つのURLの値を持った関数を発生して、その値を1度に関数の処理へ引き渡すことにより、それぞれのフレームに新しいウィンドウを開いています。この時に、本来のウィンドウ名に当る部分は、フレーム名として取り扱われます。
実際に試す場合には、この他にも"page1.html"〜"page3.html"の2つのHTMLファイルを用意してください。
なお、フレームを抜けてページを表示するには、「window.open(URL,"_top")」と、ウィンドウ名に「_top」を指定します。具体的な使い方は、「Formオブジェクト」の「ボタンをリンクに使う」(P.372)を参考にしてください。
また、「HREF="JavaScript:関数"」の使い方は、「Linkオブジェクト」の「リンクをボタンのように使う -1-」(P.366)を参考にしてください。

Sample

【フレーム】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<FRAMESET COLS="120,*">
    <FRAME SRC="f1.html" NAME="f1">
  <FRAMESET ROWS="50%,50%">
    <FRAME SRC="f2.html" NAME="f2">
    <FRAME SRC="page1.html" NAME="f3">
   </FRAMESET>
</FRAMESET>
```

```
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function CH1(P1,P2){
    window.open(P1,"f2");
    window.open(P2,"f3");
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<A HREF="JavaScript:CH1('page2.html','page3.html')">Cha
nge!!</A>
<P>
<A HREF="JavaScript:CH1('f2.html','page1.html')">元に戻す</A>
</BODY>
</HTML>
```

【f2.html】

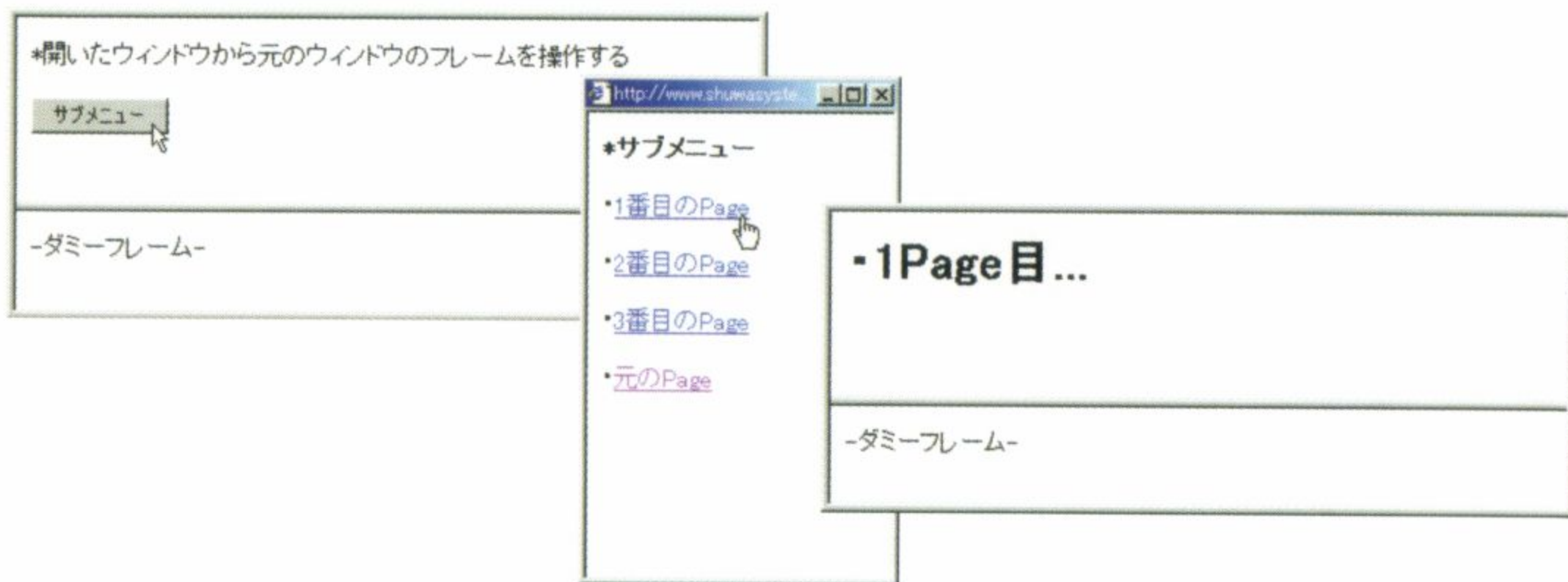
```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*複数のフレームを同時に変更する - リンクを使う -<P>
</BODY>
</HTML>
```


window.open()→「windowオブジェクト」の「新しいウィンドウを開く」:P.296参照

HREF="JavaScript:関数"→「Link・Anchorオブジェクト」の「リンクをボタンのように使う - 1 -」:P.366参照

JavaScript

開いたウィンドウから元のウィンドウのフレームを操作する

window.open()
TARGET=フレーム名

＊開いたウィンドウから元のウィンドウのフレームを操作する
サブメニュー
-ダミーフレーム-
http://www.shuwasyste...
＊サブメニュー
・1番目のPage
・2番目のPage
・3番目のPage
・元のPage
・1Page目...
-ダミーフレーム-

解説

サンプルは、ウィンドウ名とフレーム名が、同じように取り扱われることを利用しています。

フレーム内から新しいウィンドウを開き、そのウィンドウから<A HREF="URL" TARGET=フレーム名>というように、ターゲットウィンドウの設定にフレーム名を指定することにより、指定したフレームにページをロードしています。

実際に試す場合には、この他にも"page1.html"～"page3.html"の3つのHTMLファイルを用意してください。

「windowオブジェクト」の「開いたウィンドウから元のウィンドウを操作する」(P.300)は、Netscape Navigator3.0以降でしか使えませんが、このサンプルはNetscape Navigator2.0でも正常に作動します。

Sample

【フレーム部分】

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<FRAMESET ROWS="*,50">
    <FRAME SRC="f1.html" NAME="f1">
    <FRAME SRC="f2.html" NAME="f2">
</FRAMESET>
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function wopen15(){
    var WO1;
    WO1=window.open("menu1.html","WindowOpen14",
                    "toolbar=no,location=no,directories
=no,status=no,menubar=yno,scrollbars=no,resizable=no,
width=200,height=300");
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*開いたウィンドウから元のウィンドウのフレームを操作する<P>
<FORM NAME="MENU1">
<input TYPE="button" NAME= "nenu1" VALUE=" サブメニュー " on
Click="wopen15()">
</FORM>
</BODY>
</HTML>
```

【f2.html】

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
-ダミーフレーム-
</BODY>
</HTML>
```

【menu1.html】

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
<B>*サブメニュー</B>
<P>
・<A HREF="page1.html" TARGET=f1>1番目のPage</A>
<P>
・<A HREF="page2.html" TARGET=f1>2番目のPage</A>
<P>
・<A HREF="page3.html" TARGET=f1>3番目のPage</A>
<P>
・<A HREF="f1.html" TARGET=f1>元のPage</A>
<P>
</BODY>
</HTML>
```

JavaScript

文字を書き出す

document.write(文字列) [メソッド]

*文字を書き出す

例2)こうしても
例3)こうしても
例1)こうしても
結果は同じ。

解説

JavaScriptでブラウザに文字を書き出す時には、「write()」メソッドを使用します。その時に、JavaScript内にHTMLのタグを書けば、HTMLで普通に記述した時と同じようにタグが評価され、書き出された文字は、普通のテキスト文と同様に前後に挟まれたタグを評価します。
また、JavaScript自体も、HTMLのタグと同じように文字を修飾するコマンドを多数持っています。
JavaScriptが持っている文字を修飾するメソッドに関しては、「stringオブジェクト」(P.476〜)を参照してください。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*文字を書き出す<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("<B>例2)こうしても</B>")
//--->
</SCRIPT>
<BR>
<B>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("例3)こうしても")
//--->
</SCRIPT>
</B>
<BR>
```

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("例1)こうしても".bold())
//--->
</SCRIPT>
<BR>
結果は同じ。
</BODY>
</HTML>
```

bold()→「stringオブジェクト」の「太字(ボールド)にする」:P.479参照

重要

長い文章を書き出したい時は

「document.write()」などを使って長い文章を書き出した時、ブラウザのバージョンによっては、エラーが発生することがあります。

これは、Netscape Navigatorが1行の内に1バイト文字で255字まで、2バイト文字だとその半分の文字しか扱えないことからくる問題です。

この問題を回避するために、「document.write("文字列A"+"文字列B")」といった具合に、JavaScriptで書き出す文章は短く分けることをお勧めします。

JavaScript

改行付きで文字を書き出す

document.writeln(*文字列*) [メソッド]

*改行付きで文字を書き出す

このScriptはこのように<PRE></PRE>タグ
内で文章を改行する時に使います。

解説

「writeln()」メソッドは、コマンドの終了位置に改行コードを付けて文字を書き出します。
<PRE>タグ内でのみで意味を持ちます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*改行付きで文字を書き出す<P>
<PRE>
<SCRIPT language="JavaScript">
<!---
document.writeln("このScriptはこのように&lt;PRE&gt;&lt;/PRE&gt;タグ");
document.writeln("内で文章を改行する時に使います。");
//--->
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

<PRE>→「スタイルとレイアウト」の「空白や改行を入力した通りに表示させる」:P.69参照

ドキュメントの情報を取得する

document.title [プロパティ]
document.URL [プロパティ]
document.referrer [プロパティ]

*ドキュメントの情報を取得する

タイトル:Title
URL:file:///C|/WINDOWS/????????/0823/2/06_03/03doc.html
リンク元のURL:file:///C|/WINDOWS/????????/0823/2/06_03/link.html

解説

documentオブジェクトが記述してある、HTMLファイルの情報を取得するプロパティです。
「title」プロパティはHTMLの<TITLE>部分を、「URL」プロパティはHTMLファイル自身のURLを、「referrer」プロパティはHTMLファイルがリンクされていたURLを、それぞれ返します。
「referrer」プロパティは、Internet Explorer3.xでは、リンク元のURLではなく、自分自身のURLを表示してしまいます。この問題は、Internet Explorer4.xやMacintosh版Internet Explorer3.01では解消されています。しかし、Internet Explorer5.0でもリンク元URLを取得できない場合がありますので注意してください。
これらのプロパティは読み出し専用です。

Sample

```
<HTML>
<HEAD>
<TITLE>Title</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ドキュメントの情報を取得する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("タイトル:",document.title);
document.write("<BR>");
document.write("URL:",document.URL);
document.write("<BR>");
document.write("リンク元のURL:",document.referrer);
//--->
</SCRIPT>
</BODY>
</HTML>
```

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

ファイルの更新日時を取得する

document.lastModified 【プロパティ】

```
*ファイルの更新日時を取得する

Last update:07/08/99 12:37:40
```

解説

「lastModified」プロパティは、documentオブジェクトが記述してあるHTMLファイルの、最終更新日時を返します。
この日時は、HTMLファイルが置かれているHTTPサーバーのタイムスタンプが参照されます。HTTPサーバーは国際標準時で運用されていることが多く、ファイルの最終更新日時が日本時間とずれる場合があります。
このプロパティは読み出し専用です。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ファイルの更新日時を取得する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("Last update:",document.lastModified)
//--->
</SCRIPT>
</BODY>
</HTML>
```

重要

MacintoshでHTTPサーバーを運用する時の注意

「lastModified」プロパティは、どのバージョンのNetscape NavigatorもMacintoshからは正確なファイル更新日を取得できません。Macintoshを使ってHTTPサーバーを運用している場合は、注意が必要です。

開いたウィンドウに文字を記述する

`document.write(文字列)` ［メソッド］
`document.open()` ［メソッド］
`document.close()` ［メソッド］

Before

*開いたウィンドウに文字を記述する

Open!!

After

イメージも表示できます。

Window Close

JavaScriptも記述できます。

解説

サンプルでは、URLの指定なしに「window.open()」を実行することにより、何も記述していないウィンドウを開き、その中にJavaScriptでドキュメントを記述しています。
ウィンドウへの記述方法は、まず、「document.open()」でドキュメントストリームを開き、そこへ「document.write()」でドキュメントを書き出します。そして、ドキュメントの記述を終らせる時は、「document.close()」でドキュメントストリームを閉じます。
「document.write()」で書き出された文字は、HTMLのタグやJavaScriptを通常通りに評価するので、画像を貼り込んだり、JavaScriptを記述したりできます。
しかし、Netscape Navigator2.0では、画像ファイルやリンクのURLを絶対パスで指定しなければ、画像ファイルやリンクが探せない時があり、その結果、画像が表示されなかったり、リンクが機能しない場合があります。そのような場合は、画像ファイルやリンクのURLを絶対パスで指定するようにしてください。
「document.open()」は省略可能ですが、「document.close()」は必ず記述して、明示的にドキュメントストリームを閉じるようにしてください。
もし「document.close()」が記述されていない場合、JavaScriptは、最後の「document.write()」が読み込まれても、ドキュメントがまだ後に続くものと判断して、待機状態のままになります。その結果、Netscape Navigatorでは最後の行が表示されなかったり、Internet Explorerで読み込み中を示す「e」のアイコンが回りっぱなしになり、最悪の場合は何も表示されない、という状態になります。Netscape Navigatorの場合は、最後の行に
などの改行タグを入れることによって最後の行を表示させることが可能ですが、本質的な解決ではありません。

また、Internet Explorer3.0や4.0では、ブラウザの細かいバージョンによっては、うまく実行できない場合があるので注意してください。

Sample

```
<HTML>
<HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function DWO1(){
     var DW1;
     DW1=window.open("","DocWin1");
     DW1.document.open();
     DW1.document.write("<HTML><HEAD><TITLE>DOCUMENT_SAM
PLE</TITLE>");
    DW1.document.write("<"+"SCRIPT LANGUAGE='JavaScript'>  ");

    DW1.document.write("function WClose2(){                ");
    DW1.document.write("                window.close() }   ");
    DW1.document.write("</SCRIPT>                          ");
    DW1.document.write("</HEAD>                            ");
    DW1.document.write("<BODY BGCOLOR='#FFFFFF'>           ");
    DW1.document.write("<IMG NAME='kao' SRC='image.gif' ALT=
'image.gif' WIDTH='100' HEIGHT='100'> ");
    DW1.document.write("<BR>                               ");
    DW1.document.write("イメージも表¥示¥できます。          ");
    DW1.document.write("<P>                                ");
    DW1.document.write("<HR>                               ");
    DW1.document.write("<FORM>                             ");
    DW1.document.write("<INPUT TYPE='button' NAME= 'Wcl2' VAL
UE='Window Close' onClick='WClose2()'>");
    DW1.document.write("</FORM>                            ");
    DW1.document.write("<BR>                               ");
    DW1.document.write("JavaScriptも記述できます。          ");
    DW1.document.write("</BODY>                            ");
    DW1.document.write("</HTML>                            ");
    DW1.document.close();
                }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*開いたウィンドウに文字を記述する<P>
<FORM>
<INPUT TYPE="button" NAME= "dwo1" VALUE="Open!!" onClick
="DWO1()">
</FORM>
</BODY>
</HTML>
```

window.open()→「windowオブジェクト」の「新しいウィンドウを開く」:P.296参照
window.close()→「windowオブジェクト」の「ウィンドウを閉じる」:P.298参照

ドキュメントや画像を後から開く

document.write(*文字列*) 【メソッド】
document.open() 【メソッド】
document.close() 【メソッド】

Before

*ドキュメントや画像を後から開く

Open!!

After

このように、イメージを開くこともできます。

解説

新しいウィンドウに文字を記述するのと同じ要領で、ブラウザに表示されている文字を書き替えることができます。
サンプルでは、「Open!!」ボタンがクリックされると、「document.open()」でドキュメントストリームが開かれます。同時に、現在表示されている文字やフォームなどが消され、その後に「document.write()」で文字が書き出されます。
ウィンドウに文字を記述した時と同様、「document.close()」は必ず入れるようにしてください。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function Doc4(){
    document.open();
    document.write("<IMG SRC='image.gif' ALT='image.gif' ↵
WIDTH='100' HEIGHT='100'>");
    document.write("<P>                    ");
    document.write("このように、イメージを開くこともできます。");
    document.close();
}
//--->
</SCRIPT>
</HEAD>
```

```
<BODY BGCOLOR="#FFFFFF">
*ドキュメントや画像を後から開く<P>
<FORM>
<INPUT TYPE="button" NAME= "DOC4" VALUE="Open!!" onClick
="Doc4()">
</FORM>
</BODY>
</HTML>
```

重要

書き出された文字が文字化けする時は

「document.write()」で文字を書き出したり、「alert()」などのダイアログボックスに文字を表示した時に、「表示」といった漢字が「侮ｦ」と文字化けしてしまう場合があります。

これは、日本語の文字コードと英語の文字コードとの間に問題があるためです。

もし、文字化けが発生した場合、上のサンプルのように「表¥示」と、文字化けしている文字の後ろに「¥」(又はバックスラッシュ)を入れます。

この他にも、「可能¥」や「予¥定」、「申¥し訳」などの割と使いそうな漢字にも同様の問題が起き、その結果エラーを起こす場合があります。

注意

Netscape2.0でドキュメントをその場で書き替えるスクリプトを使用する時の注意

Netscape Navigator2.0(Macintosh版で確認)では、ドキュメントや画像を後から開くスクリプトや、文字を消去をするスクリプトは非常に不安定です。最悪の場合システムエラーを引き起こす場合があります。

また、「document.close()」を記述すると、本来そこでドキュメントの記述が終了したと判断されなければいけないはずなのですが、反対に待機状態になってしまう場合があります。

文字を消去する

document.open() [メソッド]

Before	After
*文字を消去する 下のclearボタンを押すと、ここに 表示されている文字やボタンが消去されます。 clear	

解説

「document.open()」でドキュメントストリームが開いた時、ブラウザに表示されている文字などが消されることを利用したスクリプトです。
「clear」ボタンが押されると「document.open()」が実行され、表示されている文字が消去されます。
JavaScript1.0には、ドキュメントを消去するために「clear()」というメソッドが用意されていましたが、JavaScript1.1以降では、「clear()」メソッドは削除されています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function Dclear(){ document.open() }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<B>*文字を消去する</B>
<P>
下のclearボタンを押すと、ここに
<BR>
表示されている文字やボタンが消去されます。
<P><HR>
<FORM>
<INPUT TYPE="button" NAME= "Dcl" VALUE=" clear " onClick
="Dclear()">
</FORM>
</BODY>
</HTML>
```

JavaScript

テキストやリンクの色を指定する

document.alinkColor="*色指定*" 【プロパティ】
document.bgColor="*色指定*" 【プロパティ】
document.fgColor="*色指定*" 【プロパティ】
document.linkColor="*色指定*" 【プロパティ】
document.vlinkColor="*色指定*" 【プロパティ】

Before

*テキストやリンクの色を指定する
・テキストの色は白。
・リンクの色は緑。
・すでに行ったことのあるリンクの色は黄色。
・リンクをクリックした時の色は黒。

After

*テキストやリンクの色を指定する
・テキストの色は白。
・リンクの色は緑。
・すでに行ったことのあるリンクの色は黄色。
・

解説

「alinkColor」プロパティはアクティブリンクの色の値を、「fgColor」プロパティはフォアグラウンド、つまりテキストの色の値を、「linkColor」プロパティはリンクの色の値を、「vlinkColor」プロパティは、すでに行ったことのあるリンクの色の値を、それぞれ持っています。
サンプルでは、それぞれのプロパティに色の値を設定することによって、ブラウザに表示されるテキストやリンクの色を指定しています。
色指定は、色の名前か16進数で設定します。
これらの設定は、<BODY>タグ内に記述する、HTMLでの色指定より優先されます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.fgColor="white";
document.linkColor="green";
document.vlinkColor="FFFF00";
document.alinkColor="000000";
//--->
```

```
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#000000">
*テキストやリンクの色を指定する
<P>
・テキストの色は白。
<BR>
・<A href="NoLink.html">リンクの色は緑。</A>
<BR>
・<A href="http://www.netscape.com/">すでに行ったことのあるリンク
の色は黄色。</A>
<BR>
・<A href="#">リンクをクリックした時の色は黒。</A>
<BR>
</BODY>
</HTML>
```

<BODY BGCOLOR="色指定">→:「ページの基礎となる内容」の「背景色を指定する」P.30参照
巻末付録「カラーチャート1〜3」:巻末参照

Tip

「window.open()」で開いたウィンドウに「document.write()」で文字を書き出す時の注意

サンプル「開いたウィンドウに文字を記述する」(P.337)のように、「window.open()」で新しく開いたウィンドウに「document.write()」で文字を書き出すスクリプトは、ウィンドウを開いてHTMLファイルを読み込むスクリプトに比べると、外部からファイルを読み込む必要がないので表示が早く、ちょっとしたメッセージを表示する場合には便利です。

しかし、このスクリプトは、Internet Explorerではうまく機能しない場合があります。特にサンプルのように、書き出す文字にJavaScriptなどのスクリプトが混じっているような時は、エラーになる場合が多くあります。また、同じバージョンのInternet Explorerでも、より新しいものではエラーが出なくても、古いものではエラーになる場合があります。

「開いたウィンドウに文字を記述する」は、チェックの結果、一部のInternet Explorerに作動の不安定な部分があったので、基本的にInternet Explorer3.0〜4.0には対応していないことにしています。けれども、書き出す内容や環境によっては、正常に作動します。反対にいえば、自分の環境では正常に動作するように見えるスクリプトでも、他のInternet Explorerではエラーになる場合があるということですので、十分にテストを行うようにしてください。

JavaScript

背景色を変えるボタンを作る

document.bgColor="*色指定*" [プロパティ]

*背景色を変えるボタンを作る

バックを白へ　バックを緑へ　バックを黄色へ　バックを黒

解説

「bgColor」プロパティは、バックグラウンドの色の値を持っています。
サンプルでは、ボタンをクリックした時に、「document.bgColor」プロパティで設定している色の値が評価され、背景色がその場で変わります。

Sample

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*背景色を変えるボタンを作る<P>
<HR>
<FORM>
 <INPUT TYPE="button" NAME= "BgColor1" VALUE="バックを白へ" 
onClick="document.bgColor='white'">
 <INPUT TYPE="button" NAME= "BgColor2" VALUE="バックを緑へ" 
onClick="document.bgColor='green'">
 <INPUT TYPE="button" NAME= "BgColor3" VALUE="バックを黄色へ
" onClick="document.bgColor='#FFFF00'">
 <INPUT TYPE="button" NAME= "BgColor4" VALUE="バックを黒" on
Click="document.bgColor='#000000'">
</FORM>
</BODY>
</HTML>
```

巻末付録「カラーチャート1〜3」:巻末参照

テキストの色を変えるボタンを作る

document.fgColor="色指定" [プロパティ]

IE3.0 IE4.0 IE5.0

*テキストの色を変えるボタンを作る
Explorerでは、レイアウト確定後も文字の色を変えることができます。
文字を白へ 文字を緑へ 文字を黄色へ 文字を黒へ

解説 Internet Explorerでは、ページのレイアウトが確定した後からでも、表示されているテキストの色を変えることができます。
サンプルでは、ボタンをクリックされた時、ブラウザで表示されているテキストの色の値を持った「document.fgColor」プロパティで設定している色が評価され、テキストの色がその場で変わります。

Sample

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*テキストの色を変えるボタンを作る<P>
Explorerでは、レイアウト確定後も文字の色を変えることができます。
<HR>
<FORM>
 <INPUT TYPE="button" NAME= "FgColor1" VALUE="文字を白へ" on
Click="document.fgColor='white'">
 <INPUT TYPE="button" NAME= "FgColor2" VALUE="文字を緑へ" on
Click="document.fgColor='green'">
 <INPUT TYPE="button" NAME= "FgColor3" VALUE="文字を黄色へ"
onClick="document.fgColor='#FFFF00'">
 <INPUT TYPE="button" NAME= "FgColor4" VALUE="文字を黒へ" on
Click="document.fgColor='#000000'">
</FORM>
</BODY>
</HTML>
```

parent→「frameオブジェクト」の「入力されたURLを別フレームに表示する」:P.324参照
巻末付録「カラーチャート1～3」:巻末参照

JavaScript

テキストの色を変える

document.write(文字列) [メソッド]

*テキストの色を変える
Netscapeの場合一度レイアウトされた文字の
色を変えることができません。
ちょっとした工夫が必要になります。

テキストを白へ テキストを緑へ
テキストを黄色へ

テキストを白へ テキストを緑へ
テキストを黄色へ

*テキストの色を変える
Netscapeの場合一度レイアウトされた文字の
色を変えることができません。
ちょっとした工夫が必要になります。

テキストを白へ テキストを緑へ
テキストを黄色へ

*テキストの色を変える
Netscapeの場合一度レイアウトされた文字の
色を変えることができません。
ちょっとした工夫が必要になります。

テキストを白へ テキストを緑へ
テキストを黄色へ テキストを黒へ

解説

Netscape Navigatorでは、1度レイアウトが確定してしまうと、バックグラウンドの色以外のテキストやリンクの色を変更することができません。
サンプルでは、フレーム「f2」からボタンがクリックされた時に、テキストの色の値と一緒にフレーム「f1」へ、フレーム「f1」と同じレイアウトのドキュメントを書き出しています。

Sample

【フレーム】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<FRAMESET ROWS="*,100">
    <FRAME SRC="f1.html" NAME="f1" >
    <FRAME SRC="f2.html" NAME="f2">
</FRAMESET>
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript

【f1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<HEAD>
<BODY BGCOLOR="#FFFFFF">
*テキストの色を変える<BR>
Netscapeの場合一度レイアウトされた文字の<BR>
色を変えることができません。<BR>
ちょっとした工夫が必要になります。
</BODY>
</HTML>
```

【f2.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function TP(TC) {
    parent.f1.document.open()
    parent.f1.document.write("<BODY BGCOLOR='#FFFFFF' TEXT=");
    parent.f1.document.write(TC);
    parent.f1.document.write(">");
    parent.f1.document.write("*テキストの色を変える<BR>");
    parent.f1.document.write("Netscapeの場合一度レイアウトされた文字の<BR>");
    parent.f1.document.write("色を変えることができません。<BR>");
    parent.f1.document.write("ちょっとした工夫が必要になります。");
    parent.f1.document.close()
}
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<FORM>
 <INPUT TYPE="button" NAME= "TextColor1" VALUE="テキストを白へ" onClick="TP('white')">
 <INPUT TYPE="button" NAME= "TextColor2" VALUE="テキストを緑へ" onClick="TP('green')">
 <INPUT TYPE="button" NAME= "TextColor3" VALUE="テキストを黄色へ" onClick="TP('#FFFF00')">
 <INPUT TYPE="button" NAME= "TextColor4" VALUE="テキストを黒へ" onClick="TP('#000000')">
</FORM>
</BODY>
</HTML>
```

parent→「frameオブジェクト」の「入力されたURLを別フレームに表示する」:P.324参照

選択した文字を返す

JavaScript

getSelection() [メソッド]

Before

After

解説

「getSelection()」メソッドは、マウスなどで選択された文字を返します。
フォームに値を渡すことが可能なので、サンプルのように、選択した文字をテキストエリアに書き出すこともできます。
JavaScript1.2で追加されたメソッドです。

Sample

```
<SCRIPT Language="JavaScript1.2">
<!--
function selct() {
     document.OUTP.outp.value = document.getSelection()
}
// -->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*選択した文字を返す<P>
<BLOCKQUOTE>
<B>"JavaScript"</B>とは、Netscape社がWebページの処理能力を高めるために開発した <B>"LiveScript"</B> を元に、Netscape 社と Sun が共同で開発した<B>Script言語</B>で、現在Netscape Navigator 2.0以降のブラウザとInternet Explorer 3.0以降のブラウザで対応されています。
</BLOCKQUOTE>
<HR>
<FORM NAME="OUTP">
<TEXTAREA NAME="outp" ROWS=5 COLS=30>
</TEXTAREA>
<P>
```

```
<INPUT TYPE="button" VALUE="選択!!" onClick="selct()">
<INPUT TYPE=reset VALUE="Clear">
</FORM>
```


JavaScript

その他のdocumentオブジェクト

cookie 【プロパティ】

解説

JavaScriptでは、cookieファイルに情報を書き込み、利用することができます。
cookieファイルはローカルディスク上に置かれ、個人を特定する情報や、特定サイトを利用するにあたって必要な情報を保存しているので、そのユーザーがサイトを訪れた回数を表示したり、ユーザーに合わせた情報を表示することができます。
cookieファイルは、JavaScriptがアクセスできる、唯一のローカル資源です。
cookieファイルの使用については、記録できる情報量を制限するなどの対策がとられているのですが、CGIなどで利用した場合に「個人情報を取得できる」などの理由から、嫌う人もいます。

用法

```
document.cookie
```

domain 【プロパティ】

複数のサーバ名を、1つのドメイン名として取り扱えます。
例えば、1つのウインドウの別々のフレームに、「search.hamba.com」と「www.hamba.com」の2つのサーバーからのデータを表示させる場合、domainプロパティによって各々のサーバ名を「hamba.com」として扱うことができます。
ただし、ドメイン名部分を変えることはできません。つまり「search.hamba.com」を「search.com」のように変えることはできません。また、1度ドメイン名を変更すると、変更前のドメイン名を参照できなくなります。
JavaScript1.1で追加されたプロパティですが、Internet Explorerは対応していません。
そのウインドウが、他のウインドウ、あるいはフレームから、プロパティなどのデータの参照を許している状態(taintingの状態)かどうかを調べます。
実際の指定は、次の例のように行います。

用法

```
document.domain="指定するドメイン名"
```

JavaScript

戻るボタンを作る

history.back() [メソッド]

Before

*戻るボタンを作る

Back

After

Shuwa System

解説

サンプルは、ブラウザの「戻る」(Back)ボタンと同じ働きをするスクリプトです。フォームのボタンが押された時に、「onClick」で指定している「history.back()」が評価され、1つ前のページへ戻ります。戻るページが来歴内にない場合は、何も起きません。

複数のページからリンクを張っているページで使うと効果的です。

Sample

```
<FORM>
<INPUT TYPE="button" VALUE=" Back " onClick="history.
back()">
</FORM>
```

JavaScript

進むボタンを作る

history.forward() [メソッド]

Before

*進むボタンを作る

Forward

After

Shuwa System

解説

サンプルは、ブラウザの「進む」(Forward)ボタンと同じ働きをするスクリプトです。フォームのボタンが押された時に、「onClick」で指定している「history.forward()」が評価され、1つ先のページへ進みます。進むページが来歴内にない場合は、何も起きません。

Sample

```
<INPUT TYPE="button" VALUE=" Forward " onClick="history.
forward()">
</FORM>
```

複数ページを進んだり戻ったりするボタンを作る

history.go(*n*) **［メソッド］**

*複数ページを進んだり戻ったりするボタンを作る

3ページ戻る　2ページ戻る　2ページ進む　3ページ進む

解説

サンプルは、複数のページを進んだり、戻ったりするボタンのスクリプトです。
フォームのボタンが押された時に「onClick」で指定している「history.go(n)」が評価され、「n」の数値分ページを移動します。
また、「go(URL)」と指定すると、来歴内の指定されたページを表示します。移動するページが来歴内にない場合は、何も起きません。
Internet Explorerでは、うまく機能しない場合があります。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*複数ページを進んだり戻ったりするボタンを作る<P>
<FORM>
<INPUT TYPE="button" VALUE=" 3ページ戻る " onClick="history.
go(-3)">
<INPUT TYPE="button" VALUE=" 2ページ戻る " onClick="history.
go(-2)">
<INPUT TYPE="button" VALUE=" 2ページ進む " onClick="history.
go(2)">
<INPUT TYPE="button" VALUE=" 3ページ進む " onClick="history.
go(3)">
</FORM>
</BODY>
</HTML>
```

onClick→リファレンス「イベントハンドラ」の「onClick」:P.547参照

その他のhistoryオブジェクト

current 【プロパティ】

解説

「current」プロパティは、現在表示されているウインドウと同じURLの値を持っています。そのウインドウが、他のウインドウ、あるいはフレームから、プロパティの参照を許している状態(taintingの状態)の時のみ機能します。
次の例では、「current 」プロパティに「hamba.com」という値が含まれているかどうかを検索し、含まれている場合は「処理」を実行します。
JavaScript1.1で追加されたプロパティです。

用法

```
if (history.current.indexOf("hamba.com") != -1) { 処理 }
```

next 【プロパティ】

解説

「next」プロパティは、ブラウザの「進む」(Forward)ボタンで得られるのと同じURLの値を持っています。そのウインドウが、他のウインドウ、あるいはフレームから、プロパティの参照を許している状態(taintingの状態)の時のみ機能します。
次の例では、「next」プロパティに「hamba.com」という値が含まれているかどうかを検索し、含まれている場合は「処理」を実行します。
JavaScript1.1で追加されたプロパティです。

用法

```
if (history.next.indexOf("hamba.com") != -1) { 処理 }
```

previous 【プロパティ】

解説

「previous」プロパティは、ブラウザの「戻る」(Back)ボタンで得られるのと同じURLの値を持っています。そのウインドウが、他のウインドウ、あるいはフレームから、プロパティの参照を許している状態(taintingの状態)の時のみ機能します。
次の例では、「previous」プロパティに「hamba.com」という値が含まれているかどうかを検索し、含まれている場合は「処理」を実行します。
JavaScript1.1で追加されたプロパティです。

用法

```
if (history.previous.indexOf("hamba.com") != -1) { 処理 }
```

JavaScript

自ページのURLを取得する

location.href [プロパティ]
location.protocol [プロパティ]
location.hostname [プロパティ]
location.pathname [プロパティ]
location.port [プロパティ]
location.host [プロパティ]

*自ページのURLを取得する

URL:file://Pro_server/souko/urata/2-ナビオブジェ-ToU/08_01/01loc.html
プロトコル:file:
ホストコンピュータ名:
パス名:/¥¥Pro_server¥souko¥urata¥2-ナビオブジェ-ToU¥08_01¥01loc.ht
コミュニケーションポート番号:
ホスト名・ポート番号:

解説

locationオブジェクトには、URLに関する情報が格納されています。
「href」プロパティはURL全体の値を、「protocol」プロパティはURL内のhttpやftpなどのプロトコル部分の値を、「hostname」プロパティはURLの内のホスト名部分の値を、「pathname」プロパティはURLの内パス名部分の値を、「port」はURL内の:8080などのポート番号の値を、「host」はホスト名とポート番号部分の値を、それぞれ返します。
locationオブジェクトには、これ以外にも、「hash」(アンカー)・「search」(?で始まる問い合わせ文字列)プロパティがあります。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("URL:",location.href);
document.write("<BR>");
document.write("プロトコル:",location.protocol);
document.write("<BR>");
document.write("ホストコンピュータ名:",location.hostname);
document.write("<BR>");
document.write("パス名:",location.pathname);
document.write("<BR>");
document.write("コミュニケーションポート番号:",location.port);
document.write("<BR>");
document.write(" ホスト名・ポート番号:",location.host);
//--->
</SCRIPT>
```

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript

入力されたURLへ進むフォームを作る

location.href [プロパティ]

Before

*入力されたURLへ進むフォームを作る

http://www.shuwasystem.co.jp/ Go!!

After

Shuwa System

解説

サンプルでは、locationオブジェクトが動的にURLを変更できることを利用して、ボタンがクリックされたタイミングでフォームの内容を参照し、フォームに入力されたURLの値を「href」プロパティに設定することによって、フォームに入力されたURLがブラウザにロードされます。
また、もしフォームに何も入力されていない時には、警告用のダイアログボックスが開きます。入力されたURLが不正な場合は、ブラウザ自身が警告用のダイアログボックスを出して、利用者に注意をうながします。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function LC(go){
        if (go.url.value != "") { location.href=go.url.
value }
            else { alert("URLを入力してください") }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*入力されたURLへ進むフォームを作る<P>
<FORM NAME="URL">
<INPUT ="text" NAME="url" VALUE="http://" SIZE=40 >
<INPUT TYPE="button" NAME= "CF" VALUE=" Go!! " onClick="
LC(this.form)">
</FORM>
</BODY></HTML>
```

<INPUT TYPE="button">→「Formオブジェクト」の「ボタンをリンクに使う」:P.372参照

ロード完了後に次のページをロードする

location.href 【プロパティ】

Before

After

解説

サンプルでは、ページのロードが終ってから10秒後に関数「NEXT1()」が発生し、「location.href」で設定しているURLをブラウザにロードします。
次のページをロードするタイミングは、「setTimeout('NEXT1()',10000)」内の「10000」を変更することで可能です。
HTMLの<META>タグでも自動的にページをロードすることが可能ですが、その場合は回線状態によっては表示が遅くなり、ページが完全にロードされる前に次のページがロードされてしまうことがあります。このサンプルの場合は、ページが完全に読み込まれるまでイベントが発生しませんので、そのような問題を回避することができます。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT Language="JavaScript">
<!---
function NEXT1(){ location.href = "http://www.shuwasystem.
co.jp/" }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF" onLoad="setTimeout('NEXT1()',10
000)" >
*ロード完了後に次のページをロードする<P>
ページのロードが完全に終ってから10秒後に別のページへ移ります...。<P>
<IMG SRC="robot.gif" ALT="robot.gif" WIDHT="474" HEIGHT
="198">
</BODY>
</HTML>
```

setTimeout()→「windowオブジェクト」の「ステータス行に文字を流す」:P.292参照

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript対応ページと未対応ページを振り分ける

location.href 【プロパティ】

・JavaScript対応のページ

*JavaScript対応ページと未対応ページを振り分ける

JavaScriptが使えないブラウザは、5秒後に"META"タグで「JavaScript未対応のページ」に飛びます。
"META"タグが使えないブラウザは、こちらからおいでください。

・JavaScript未対応のページ

解説

サンプルでは、ページのロード時に、JavaScriptに反応するブラウザは「location.href」を理解して「js.html」のページを、JavaScriptに反応しないブラウザはHTMLの<META>タグでJavaScript未対応用のページをそれぞれロードし、JavaScriptにも<META>タグにも反応しないブラウザには、リンクを設定してページの振り分けを行っています。

Sample

```
<HTML>
<HEAD>
<META http-equiv="refresh"content="5;URL=nojs.html">
<TITLE></TITLE>
<SCRIPT Language="JavaScript">
<!---
function gojs(){ location.href = "js.html" }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*JavaScript対応ページと未対応ページを振り分ける<P>
JavaScriptが使えないブラウザは、5秒後に"META"タグで「JavaScript未対応
のページ」に飛びます。<BR>
<A HREF="nojs.html">"META"タグが使えないブラウザは、こちらからおいでく
ださい。</A>
<SCRIPT Language="JavaScript">
<!---
gojs()
//--->
</SCRIPT>
</BODY></HTML>
```

<META>→:「ページの基礎となる内容」の「自動的にページを読み込む」:P.36参照

JavaScript
URL情報を利用する

各ブラウザ専用ページに振り分ける

location.href 【プロパティ】

Before

*各ブラウザ専用ページに振り分ける

ブラウザ判別中...

After

・Internet Explorer4.x用のページ

・Netscape Navigator4.x用のページ

解説

サンプルでは、navigatorオブジェクトのブラウザ名とバージョンを取得するプロパティを使用してブラウザの種類とバージョンを判断し、それぞれのブラウザ専用ページへの振り分けを行っています。
Internet Explorer3.xは、バージョンとして「2.0」を返すものがあるので、その点も考慮しています。またInternet Explorer5.0は、現在バージョンとして「4.0」を返すので、バージョン情報の中から「MSIE 5」という文字列を検索して振り分けています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function GoPege() {
    if( navigator.appName.charAt(0)=="N" ){
        if(navigator.appVersion.charAt(0)==2){ location.
href = "NN2.html" }
        if(navigator.appVersion.charAt(0)==3){ location.
href = "NN3.html" }
        if(navigator.appVersion.charAt(0)==4){ location.
href = "NN4.html" }
    }
    if( navigator.appName.charAt(0)=="M" ){
        if(navigator.appVersion.charAt(0)==2){ location.
href = "IE3.html" }
        if(navigator.appVersion.charAt(0)==3){ location.
href = "IE3.html" }
```

```
        if(navigator.appVersion.charAt(0)==4){
            if (navigator.appVersion.indexOf("MSIE 5") !
= -1) { location.href = "IE5.html" }
            else { location.href = "IE4.html" }
        }
        if(navigator.appVersion.charAt(0)==5){ location.
href = "IE5.html" }
    }
}
//--->
</SCRIPT>
</HEAD>
<BODY>
*各ブラウザ専用ページに振り分ける<P>
ブラウザ判別中...
<SCRIPT LANGUAGE="JavaScript">
<!---
GoPege()
//--->
</SCRIPT>
</BODY>
</HTML>
```

---

navigator.appName→「navigatorオブジェクト」の「ブラウザ名を取得する」:P.264参照
navigator.appVersion→「navigatorオブジェクト」の「ブラウザのバージョンを取得する」:P.265参照
charAt()→「stringオブジェクト」の「n番目の文字を抜き出す」:P.487参照
indexOf()→「stringオブジェクト」の「先頭から文字列を検索する」:P.490参照

アンカーを設定する

onMouseOver [イベントハンドラ]
location.hash [プロパティ]

Before

*アンカーを設定する

この文字の上にマウスポインタを乗せると...

After

ここに来ます!!

解説

サンプルでは、リンクの上にマウスポインタが乗った時、関数「LinkMo4('#Go')」が発生して「location.hash」に「#Go」の値が渡され、アンカーで指定された場所にジャンプします。
Windows版のブラウザでは、スクロールバーが出ている範囲でしかページが動きません。サンプルでは、移動する範囲を広げるため、ダミーとして
タグを入れています。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function LinkMo4(go) { location.hash = go }
//-->
</SCRIPT>
</HEAD>
<BODY>
*アンカーを設定する<P>
<A href="LINK5.html" onMouseOver="LinkMo4('#Go')">この文字の上にマウスポインタを乗せると...</A><HR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR>
<HR><A NAME="#Go">ここに来ます!!</A>
<BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR>
</BODY></HTML>
```

JavaScript

リロードボタンを作る

location.reload() [メソッド]

*リロードボタンを作る

Reload

解説

サンプルは、ブラウザの「更新」(Reload)ボタンと同じ働きをするスクリプトです。ボタンがクリックされたタイミングで「reload()」が呼ばれ、ページをリロードします。JavaScript1.1から追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*リロードボタンを作る<P>
<FORM>
<INPUT TYPE="button" VALUE=" Reload " onClick="location.
reload()">
</FORM>
</BODY>
</HTML>
```

<INPUT TYPE="button">→「Formオブジェクト」の「ボタンをリンクに使う」:P.372参照

元のページへ戻れないようにする

location.replace() [メソッド]

NN3.0 NN4.0 NN4.06 IE4.0 IE5.0

Before

*元のページへ戻れないようにする

Bye!!

After

Shuwa System

解説

「replace()」メソッドは、現在表示されているURLを、「()」内で指定したURLに置き換えます。
したがって、元のページURLが来歴上に残りませんので、バックボタンを使って元のページへ戻って来ることができなくなります。
JavaScript1.1から追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*元のページへ戻れないようにする<P>
<FORM>
<INPUT TYPE="button" VALUE=" Bye!! " onClick="window.
location.replace('http://www.bekkoame.or.jp/~hamba/')">
</FORM>
</BODY>
</HTML>
```

<INPUT TYPE="button">→「Formオブジェクト」の「ボタンをリンクに使う」:P.372参照

JavaScript

リンクのURL情報を表示する

document.links[インデックス].href	[プロパティ]
document.links[インデックス].protocol	[プロパティ]
document.links[インデックス].hostname	[プロパティ]
document.links[インデックス].pathname	[プロパティ]
document.links[インデックス].port	[プロパティ]
document.links[インデックス].host	[プロパティ]
document.links[インデックス].search	[プロパティ]
document.links[インデックス].target	[プロパティ]

*リンクのURL情報を表示する

Netscape社のJavaScript Developer Central

URL:http://developer.netscape.com/tech/javascript/index.html
プロトコル:http:
ホスト名:developer.netscape.com
パス名:tech/javascript/index.html
コミュニケーションポート番号:80
ホスト名・ポート番号:developer.netscape.com:80
?以降の文字:
ターゲットウィンドウ:_Open

解説

Linkオブジェクトは、ページ上のリンクの[0]から始まる配列を作成し、その中には各リンクの情報が格納されています。
サンプルでは、リンクが1つしかないので、そのリンクは「document.links[0]」で参照できます。もしも複数のリンクが存在する場合は、インデックスの部分が上から[0]、[1]、[2]…となります。
「href」プロパティはURL全体の値を、「protocol」プロパティはURLの内httpやftpなどのプロトコル部分の値を、「hostname」プロパティはURLの内のホスト名部分の値を、「pathname」プロパティはURLの内パス名部分の値を、「port」プロパティはURL内の:8080などのポート番号の値を、「host」プロパティはホスト名とポート番号部分の値を、「search」プロパティは?で始まる問い合わせ文字列を、「target」はトランジット属性を、それぞれ返します。
Linkオブジェクトには、これ以外にも、「hash」(アンカー)プロパティがあります。これらのプロパティは読み出し専用です。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*リンクのURL情報を表示する<P>
<A HREF="http://developer.netscape.com/tech/javascript/index.html" target="_Open">Netscape社のJavaScript Developer Central</A>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("URL:",document.links[0].href);
document.write("<BR>");
document.write("プロトコル:",document.links[0].protocol);
document.write("<BR>");
document.write("ホスト名:",document.links[0].hostname);
document.write("<BR>");
document.write("パス名:",document.links[0].pathname);
document.write("<BR>");
document.write("コミュニケーションポート番号:",document.links[0].port);
document.write("<BR>");
document.write("ホスト名・ポート番号:",document.links[0].host);
document.write("<BR>");
document.write("?以降の文字:",document.links[0].search);
document.write("<BR>");
document.write("ターゲットウィンドウ:",document.links[0].target);
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript リンクの上にポインタが乗るとウィンドウを開く

onMouseOver **[イベントハンドラ]**

Before

＊リンクの上にポインタが乗るとウィンドウを開く

この文字の上にマウスポインタを乗せると…

After

¥¥Pro_server¥souko¥urata¥2-ナビオブジェ-To...

ウィンドウが開きます(^_^)。

Close

解説

JavaScriptでは、リンクの中にイベントハンドラを組み込むことができます。
サンプルでは、リンクの中にイベントハンドラ「onMouseOver」を組み込み、マウスカーソルがリンクの上に乗った時に、新しいウィンドウが開くようにしています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function LinkMo1(){
    var LM1;
     LM1=window.open("","DocWin1","toolbar=no,location=
no,directories=no,width=300,height=250");
    LM1.document.write("<HTML><HEAD><TITLE></TITLE>        ");
    LM1.document.write("<BODY BGCOLOR='#FFFFFF'>           ");
```

```
    LM1.document.write("ウィンドウが開きます(^_^)。                  ");
    LM1.document.write("<HR>                                        ");
    LM1.document.write("<FORM>                                      ");
    LM1.document.write("<INPUT type='button' name= 'Wcl2'
value=' Close ' onClick='window.Close()'>");
    LM1.document.write("</FORM>                                     ");
    LM1.document.write("</BODY>                                     ");
    LM1.document.write("</HTML>                                     ");
    LM1.document.close();
            }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
* リンクの上にポインタが乗るとウィンドウを開く<P>
<A HREF="#" onMouseOver="LinkMo1()">この文字の上にマウスポインタ
を乗せると...</A>
</BODY>
</HTML>
```

onMouseOver→リファレンス「イベントハンドラ」の「onMouseOver」:P.549参照

JavaScript

リンクをボタンのように使う - 1 -

HREF="javascript:*関数*"

Before

*リンクをボタンのように使う - 1 -

この文字をクリックすると…

After

LINK_SAMPLE - Microsoft Internet Explorer

ウィンドウが開きます(^_^)。
元ページもロードしません。

Close

解説

リンクのURLを指定する部分で「javascript:関数」と指定すると、リンクをクリックしたタイミングで関数を発生させることができます。
サンプルでは、リンクをクリックすると関数「LinkMo2()」が発生して、元のページはそのままで新しいウィンドウが開きます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function LinkMo2(){
    var LM1;
    LM1=window.open("","DocWin2","toolbar=no,location=no,di
rectories=no,width=300,height=250");
```

```
    LM1.document.write("<HTML><HEAD><TITLE>LINK_SAMPLE</TI
TLE>");
    LM1.document.write("</HEAD>                          ");
    LM1.document.write("<BODY BGCOLOR='#FFFFFF'>         ");
    LM1.document.write("<BR>                             ");
    LM1.document.write("ウィンドウが開きます(^_^)。          ");
    LM1.document.write("<BR>                             ");
    LM1.document.write("元ページもロードしません。          ");
    LM1.document.write("<HR>                             ");
    LM1.document.write("<FORM>                           ");
    LM1.document.write("<INPUT type='button' name= 'Wcl2'
 value=' Close ' onClick='window.Close()'>");
    LM1.document.write("</FORM>                          ");
    LM1.document.write("</BODY>                          ");
    LM1.document.write("</HTML>                          ");
    LM1.document.close();
            }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
* リンクをボタンのように使う - 1 -<P>
<A HREF="javascript:LinkMo2()">この文字をクリックすると...</a>
</BODY>
</HTML>
```

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

JavaScript

リンク・アンカー情報を利用する

JavaScript

リンクをボタンのように使う - 2 -

onClick
return false

[イベントハンドラ]

Before

リンクをボタンのように使う - 2 -

この文字をクリックすると...

After

LINK_SAMPLE - Microsoft Internet Explorer

ウィンドウが開きます(^_^)。
元ページもロードしません。

Close

解説

リンク上で「onClick」を使用した場合、リンクをクリックした時に関数が発生するのと同時に、リンクの動作も発生してしまいます。しかし、JavaScript1.1からは「return false」を返すと、その時点で関数の処理とリンクの動作を中止できるようになりました。
サンプルでは、リンクがクリックされると関数「LinkMo4()」が評価されて、ウインドウを開く処理を行います。その後、「return false」でイベントの処理を中止するのと同時に、リンクの動作も中止するようにしているので、他のリンクへ飛ぶ動作は発生しません。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function LinkMo4(){
    var LM1;
    LM1=window.open("","DocWin4","toolbar=no,location=no,di
rectories=no,width=300,height=250");
    LM1.document.write("<HTML></HEAD><TITLE>LINK_SAMPLE</TI
TLE>");
```

```
    LM1.document.write("</HEAD>                                    ");
    LM1.document.write("<BODY  BGCOLOR='#FFFFFF'>                  ");
    LM1.document.write("<BR>                                       ");
    LM1.document.write("ウィンドウが開きます(^_^)。               ");
    LM1.document.write("<BR>                                       ");
    LM1.document.write("元ページもロードしません。              ");
    LM1.document.write("<HR>                                       ");
    LM1.document.write("<FORM>                                     ");
    LM1.document.write("<INPUT type='button' name= 'Wcl2'
 value=' Close ' onClick='window.Close()'>");
    LM1.document.write("</FORM>                                    ");
    LM1.document.write("</BODY>                                    ");
    LM1.document.write("</HTML>                                    ");
    LM1.document.close();
    return false;
          }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
リンクをボタンのように使う - 2 -<P>
<A HREF="#" onClick="return LinkMo4()">この文字をクリックすると
...</a>
</BODY>
</HTML>
```

onClick→リファレンス「イベントハンドラ」の「onClick」:P.547参照

重要

リンクでイベントハンドラを使用する時の注意

リンク上で「onClick」などのイベントハンドラを使う時には、必ずリンクの指定をしてください。

リンクの指定がされていない時にリンクがクリックされると、サーバーの設定によっては、そのHTMLファイルが納められているディレクトリの一覧が表示されてしまうような場合もあります。

Netscape Navigator3.0以上では、「return false」を使うことによって問題は解消されますが、その他のブラウザでは、「return false」は無視されます。

もしも、リンクをボタンのように使い、クリックしても他のページへ飛びたくないような時には、「A HREF="#" 」といったようにリンクに「#」を設定するとよいでしょう。

JavaScript

ラジオボタンをリンクに使う

<INPUT TYPE="radio" NAME= " ラジオオブジェクト名 " VALUE=" 値 " イベントハンドラ>

Topへ
1Page目
2Page目
3Page目
フレームを抜ける
*ラジオボタンをリンクに使う
・1Page目…
・2Page目…
・3Page目…
・フレームを抜ける

解説

サンプルでは、ラジオボタンにイベントハンドラ「onClick」を設定することにより、ラジオボタンがクリックされると関数が発生し、URLが引き渡されて、フレーム名「f2」にページがロードされます。
locationオブジェクトの「href」プロパティを使ってフレームにページを読み込む場合、フレームを抜けてページを表示するには、サンプルのように「parent.top.location.href=URL」と、「top」プロパティを指定します。

Sample

【フレーム】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<FRAMESET COLS="170,*">
    <FRAME SRC="f1.html" NAME="f1">
    <FRAME SRC="f2.html" NAME="f2">
</FRAMESET>
```

```
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function P1(w1) { parent.f2.location.href=w1 }
function TP(w2) { parent.top.location.href=w2 }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<FORM>
 <INPUT TYPE="radio" NAME= "FRGo" VALUE="FR" onClick="P1
('f2.html')" checked>Topへ
<P>
 <INPUT TYPE="radio" NAME= "FRGo" VALUE="FR" onClick="P1
('page1.html')">1Page目
<P>
 <INPUT TYPE="radio" NAME= "FRGo" VALUE="FR" onClick="P1
('page2.html')">2Page目
<P>
 <INPUT TYPE="radio" NAME= "FRGo" VALUE="FR" onClick="P1
('page3.html')">3Page目
<P>
 <INPUT TYPE="radio" NAME= "FRGo" VALUE="FR"  onClick="
TP('top.html')">フレームを抜ける
</FORM>
</BODY>
</HTML>
```

【f2.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ラジオボタンをリンクに使う
</BODY>
</HTML>
```


<INPUT TYPE="radio">→「入力フォーム」の「ラジオボタンを作る」:P.155参照

parent→「frameオブジェクト」の「入力されたURLを別フレームに表示する」:P.324参照

parent.フレーム名.location.href=URL→「frameオブジェクト」の「複数のフレームを同時に変更する - ボタンを使う -」:P.326参照

JavaScript

ボタンをリンクに使う

<INPUT TYPE="button" NAME= "*ボタンオブジェクト名*" VALUE="*値*" *イベントハンドラ* >

＊ボタンをリンクに使う

Topへ　1Page目　2Page目　3Page目　フレームを抜ける

・1Page目…

Topへ　1Page目　2Page目　3Page目　フレームを抜ける

・2Page目…

Topへ　1Page目　2Page目　3Page目　フレームを抜ける

・3Page目…

Topへ　1Page目　2Page目　3Page目　フレームを抜ける

・フレームを抜ける

解説

サンプルでは、ボタンにイベントハンドラ「onClick」を設定することにより、ボタンがクリックされると関数が発生し、URLが引き渡されて、フレーム名「f1」にページがロードされます。
windowオブジェクトの「open()」メソッドを使ってフレームにページを読み込む場合、フレームを抜けてページを表示するには、サンプルのように「window.open(URL,"_top")」と、ウィンドウ名に「_top」を指定します。

Sample

【フレーム】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
```

```
<FRAMESET ROWS="*,100">
    <FRAME SRC="f1.html" NAME="f1">
    <FRAME SRC="f2.html" NAME="f2">
</FRAMESET>
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ボタンをリンクに使う
</BODY>
</HTML>
```

【f2.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function P1(w1) { window.open(w1,"f1") }
function TP(w2) { window.open(w2,"_top") }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<FORM>
 <INPUT TYPE="button" NAME= "FBGo1" VALUE="Topへ" onClick
="P1('f1.html')">
 <INPUT TYPE="button" NAME= "FBGo2" VALUE="1Page目" on
Click="P1('page1.html')">
 <INPUT TYPE="button" NAME= "FBGo3" VALUE="2Page目" on
Click="P1('page2.html')">
 <INPUT TYPE="button" NAME= "FBGo4" VALUE="3Page目" on
Click="P1('page3.html')">
 <INPUT TYPE="button" NAME= "FBGo4" VALUE="フレームを抜ける
" onClick="TP('top.html')">
</FORM>
</BODY>
</HTML>
```

<INPUT TYPE="button">→「入力フォーム」の「スクリプトで利用するボタンを作る」:P.154参照

parent→「frameオブジェクト」の「入力されたURLを別フレームに表示する」:P.324参照

window.open("URL","フレーム名") →「frameオブジェクト」の「複数のフレームを同時に変更する - リンクを使う -」:P.328参照

JavaScript

メニューをリンクに使う

<SELECT NAME="セレクトオブジェクト名" イベントハンドラ >
<OPTION>文字列
フォームオブジェクト名.セレクトオブジェクト名.selectedIndex
onChange 【イベントハンドラ】

*メニューをリンクに使う
1Page目
Topへ
1Page目
2Page目
3Page目
フレームを抜ける
・1Page目…
・2Page目…
・3Page目…
・フレームを抜ける

解説

サンプルでは、フォームの内容に変化があった時のイベントを取得して処理を発生するイベントハンドラ「onChange」によって、メニューのオプションが変更された時に、関数「FC()」の処理が発生します。
JavaScriptは、自動的に0から始まるオプションの配列を作成しているので、どのオプションが選ばれたかはそれで参照することができます。具体的には、「WO.FSGo.selectedIndex == 0」は、セレクト名「FSGo」のインデックス1番目、つまり「<OPTION> Topへ」を指します。この値が真(true)の時は、そこで指定しているURL「f1.html」がフレーム「f1」に読み込まれます。

Sample

【フレーム】

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<FRAMESET ROWS="*,80">
    <FRAME SRC="f1.html" NAME="f1">
    <FRAME SRC="f2.html" NAME="f2">
</FRAMESET>
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
```

```
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*メニューをリンクに使う
</BODY>
</HTML>
```

【f2.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function FC(WO) {
        if (WO.FSGo.selectedIndex == 0) { parent.f1.loca
tion.href = "f1.html" }
        if (WO.FSGo.selectedIndex == 1) { parent.f1.loca
tion.href = "page1.html" }
        if (WO.FSGo.selectedIndex == 2) { parent.f1.loca
tion.href = "page2.html" }
        if (WO.FSGo.selectedIndex == 3) { parent.f1.loca
tion.href = "page3.html" }
        if (WO.FSGo.selectedIndex == 4) { parent.top.loc
ation.href = "top.html" }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<FORM>
  <SELECT NAME="FSGo" onChange="FC(this.form)">
    <OPTION> Topへ
    <OPTION>1Page目
    <OPTION>2Page目
    <OPTION>3Page目
    <OPTION> フレームを抜ける
   </SELECT>
</FORM>
</BODY>
</HTML>
```

<SELECT>→「入力フォーム」の「メニューを作る」:P.157参照

parent→「frameオブジェクト」の「入力されたURLを別フレームに表示する」:P.324参照

parent.フレーム名.location.href=URL→「frameオブジェクト」の「複数のフレームを同時に変更する - ボタンを使う -」:P.326参照

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

フォームに文字を流す

JavaScript

```
<FORM NAME="フォームオブジェクト名">
<INPUT TYPE="text" NAME="テキストオブジェクト名" size=ピクセル>
document.フォーム名.テキストフォーム名.value
```

*フォームに文字を流す

ここに

*フォームに文字を流す

ここにメッセージを入れます

*フォームに文字を流す

ここにメッセージを入れます……。

解説

サンプルでは、文字をスクロールさせる部分は、「windowオブジェクト」の「ステータス行に文字を流す」(P.292)と同じです。

ただし、文字をスクロールさせるのを、ステータス行ではなくフォーム内のテキスト欄にするために、「window.status」と指示していた部分が「document.Fmess.fmess.value」となっています。これは、documentオブジェクトの中の「Fmess」という名前のフォームオブジェクト内にある、「fmess」という名前のtextオブジェクト名の値であることを表わし、そこに文字列を代入しています。

オブジェクトの階層が深くて少々ややこしく感じるかもしれませんが、常にオブジェクトの階層関係を念頭に置いてスクリプトを書くようにすればよいでしょう。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
var TC = 0 ;
var Fm1 = "                                                    ";
var Fm2 = "ここにメッセージを入れます......。";
var Fm = Fm1+Fm2;
```

```
function FMess() {
    if (TC < 1000) { //ここの数値を変えることによってスクロールする時間
が変わります
        TC++ ;
                    document.Fmess.fmess.value = Fm;
                     Fm = Fm.substring(2,Fm.length) + Fm.
substring(0,2);
                    setTimeout("FMess()",300);
                     }
   else { document.Fmess.fmess.value = "" }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF" onLoad="FMess()">
*フォームに文字を流す<P>
<FORM NAME="Fmess">
<INPUT TYPE="text" NAME="fmess" SIZE=50>
</FORM>
</BODY>
</HTML>
```

<INPUT TYPE="text">→「入力フォーム」の「1行のテキスト入力フィールドを作る」:P.142参照

注意

それはJavaScriptのせいではないんだけど

よく聞かれる質問で、『フォームから送られるデータを「mailto:」オプションでメールで受け取るようにしているんだけど、メールで受けたフォームのデータが文字化けして読めない』というのがあります。フォーム内にJavaScriptを設定したサンプルをWebページ上で公開しているので、このような質問が来るのでしょうが、これに関してまず明確にしておくべきことは、「その現象はJavaScriptとはなんにも関係がない」ということと、「その文字は別に文字化けしているわけではない」ということです。

フォームから送られるデータは、URL形式という形式に変換して送られます。フォームから送られてくる「%」付きの文字が、これになります。したがって、受け取ったデータは、元の読めるような形式にデコードする必要があります。

デコードするツールには、「ClipDecoder」などがあります。

JavaScript

フォーム内容の変更をチェックする

onChange **[イベントハンドラ]**

Before

*フォーム内容の変更をチェックする

hamba@bekkoame.ne.jp

After

*フォーム内容の変更をチェックする

sanba@bekkoame.ne.jp

Microsoft Internet Explorer

フォームの内容が変更されました。元に戻しますか?

OK キャンセル

解説

サンプルでは、イベントハンドラ「onChange」を使って、フォームの内容に変化があったかどうかをチェックしています。
もしもテキストエリア内の値が変えられた場合、フォームを抜ける時に「onChange」が関数を発生し、テキストエリアの値を元に戻すかどうかを確認するダイアログボックスを出す処理を実行します。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function OC() {
    if ( confirm ("フォームの内容が変更されました。元に戻しますか?") ) {
        document.OnC.onc.value = "hamba@bekkoame.ne.jp";
    }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*フォーム内容の変更をチェックする
<FORM NAME="OnC">
<INPUT TYPE="text" NAME="onc" VALUE = "hamba@bekkoame.
ne.jp" SIZE=30 onChange="OC()">
</FORM>
</BODY></HTML>
```

<INPUT TYPE="text">→「入力フォーム」の「1行のテキスト入力フィールドを作る」:P.142参照
confirm() →「windowオブジェクト」の「確認ボタン付きのダイアログボックスを開く」:P.288参照

フォームの内容をチェックする

<FORM NAME="*フォームオブジェクト名*" ACTION="*送信先*" METHOD="POST" *イベントハンドラ*>
<INPUT TYPE="checkbox" NAME="*チェックボックスオブジェクト名*" VALUE="*値*" *イベントハンドラ*>
<INPUT TYPE="reset" VALUE="*値*" *イベントハンドラ*>

Before

*フォームの内容をチェックする

お名前: (必ず入力してください)

ブラウザは何をお使いですか?(必ず1つ以上チェックして下さい)
- Netscape4.x
- Netscape3.x
- Netscape2.x
- Explorer5.x
- Explorer4.x
- Explorer3.x
- その他

メールを送る　リセットする

After

Microsoft Internet Explorer
名前を入力してください
OK

Microsoft Internet Explorer
チェックボックスを、どれか一つ以上チェックしてください
OK

解説

サンプルでは、「メールを送る」ボタンが押された時に、イベントハンドラ「onSubmit」が関数の処理を発生して、フォーム内のチェックを行っています。
もしもその時に、名前を書く欄に何も記入されていなかったり、チェックボックスが1つもチェックされていない場合は、警告用のダイアログボックスを開きます。
チェックボックスは、1番始めにすべてのチェックボックスに「false」(偽)の値を代入し、チェックボックスがクリックされる度にその反対の値、「false」(偽)だったら「true」(真)、「true」(真)だったら「false」(偽)を代入することにより判定しています。また、「リセットする」ボタンが押された時は、すべてのチェックボックスに「false」(偽)の値が代入されるようにしています。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
var f1=false;
var f2=false;
var f3=false;
var f4=false;
```

```
var f5=false;
var f6=false;
var f7=false;
function Mcheck(){
    if (document.MAIL4.Namae.value=="") {
        window.alert("名前を入力してください");
        return false }
    if (f1==false&&f2==false&&f3==false&&f4==false&&f5
==false&&f6==false&&f7==false) {
        window.alert("チェックボックスを、どれか一つ以上チェックしてく
ださい");
        return false }
    return true;
}
function Frest() { f1=false;f2=false;f3=false;f4=false;
f5=false;f6=false;f7=false; }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*フォームの内容をチェックする<P>
<FORM NAME="MAIL4" ACTION="mailto:hamba@ppp.bekkoame.
ne.jp" METHOD="POST" onSubmit="return Mcheck()">
<B>お名前:</B><INPUT TYPE="text" NAME="Namae" SIZE=20>(必ず
入力してください)<BR>
<P>
<B>ブラウザは何をお使いですか?</B>(必ず1つ以上チェックして下さい)<BR>
<INPUT TYPE="checkbox" NAME="NorE1" VALUE="Netscape4.x"
onClick="f1=!f1">Netscape4.x<BR>
<INPUT TYPE="checkbox" NAME="NorE2" VALUE="Netscape3.x"
onClick="f2=!f2">Netscape3.x<BR>
<INPUT TYPE="checkbox" NAME="NorE3" VALUE="Netscape2.x"
onClick="f3=!f3">Netscape2.x<BR>
<INPUT TYPE="checkbox" NAME="NorE4" VALUE="Explorer5.x"
onClick="f4=!f4">Explorer5.x<BR>
<INPUT TYPE="checkbox" NAME="NorE5" VALUE="Explorer4.x"
onClick="f5=!f5">Explorer4.x<BR>
<INPUT TYPE="checkbox" NAME="NorE6" VALUE="Explorer3.x"
onClick="f6=!f6">Explorer3.x<BR>
<INPUT TYPE="checkbox" NAME="NorE7" VALUE="other" onClick
="f7=!f7">その他
<P>
<INPUT TYPE="submit" NAME="BOOK1" VALUE="メールを送る">
<INPUT TYPE="reset" VALUE="リセットする" onClick="Frest()">
</FORM>
</BODY>
</HTML>
```

window.alert()→「windowオブジェクト」の「警告用のダイアログボックスを開く」:P.287参照

フォームからの送信にメモを付ける

<INPUT TYPE="hidden" NAME="*ヒドゥンオブジェクト名*" VALUE="*値*">
onSubmit 【イベントハンドラ】

Before

*フォームからの送信にメモを付ける

お名前:Shuwa Tarou
E-mail:tarou@shuwasystem.co.jp

メールを送る　リセットする

After

メッセージ - 日本語 (自動選択)

ファイル(F) 編集(E) 表示(V) ツール(T) メッセージ(M) ヘルプ(H)

返信 全員へ返信 転送 印刷 削除 前へ 次へ

送信者 :
日時 : 1999年8月26日 12:54
宛先 :
件名 : Form posted from Mozilla

memo:
Aug 26 12:54:40 1999Title:隠しテキストボックスのテスト

Namae:
Shuwa Tarou

EMail:
tarou@shuwasystem.co.jp

BOOK2:
メールを送る

解説

隠しテキストボックスhiddenオブジェクトを使えば、フォームにメモ的に情報を付加することができます。
サンプルでは、「メールを送る」ボタンをクリックした時に、イベントハンドラ「onSubmit」が関数の処理を発生して、隠しテキストボックスに、「document.title」プロパティで取得したHTMLファイルの<TITLE></TITLE>部分とコメント、「now.toLocaleString()」メソッドで取得した日時を付加して送信しています。

Sample

```
<HTML>
<HEAD>
<TITLE>Title</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function Memo(){
    var now = new Date();
    var time = now.toLocaleString();
    var tit = document.title;
    var com = ":隠しテキストボックスのテスト";
    document.MAIL2.memo.value = time + tit + com;
}
//--->
</SCRIPT>
<HEAD>
<BODY BGCOLOR="#FFFFFF">
*フォームからの送信にメモを付ける<P>
<HR>
<FORM NAME="MAIL2" ACTION="mailto:hamba@ppp.bekkoame.
ne.jp" METHOD="POST" onSubmit="Memo()">
<INPUT TYPE="hidden" NAME="memo" VALUE="">
<B>お名前:</B>
<INPUT TYPE="text" NAME="Namae" SIZE=40>
<BR>
<B>E-mail:</B>
<INPUT TYPE="text" NAME="EMail" SIZE=45>
<P>
<INPUT TYPE="submit" NAME= "BOOK2" VALUE="メールを送る">
<INPUT TYPE="reset" VALUE="リセットする">
</FORM>
</BODY>
</HTML>
```

<FORM>→「入力フォーム」の「メールで送信するフォームを送る」:P.140参照
<INPUT TYPE="hidden">→「入力フォーム」の「隠しフィールドを作る」:P.146参照
document.title→「documentオブジェクト」の「ドキュメントの情報を取得する」:P.335参照
Date()→「Dateオブジェクト」の「年・月・日・時・分・秒を表示する」:P.432参照
toLocaleString()→「Dateオブジェクト」の「国際標準時やローカルタイムを表示する」:P.438参照
onSubmit→リファレンス「イベントハンドラ」の「onSubmit」:P.549参照

メール送信時に挨拶を表示する

document.フォームオブジェクト名.テキストオブジェクト名.value
document.フォームオブジェクト名.セレクトオブジェクト名.options[document.フォームオブジェクト名.セレクトオブジェクト名.selectedIndex].value
onSubmit 【イベントハンドラ】

Before

*メール送信時に挨拶を表示する
お名前:Shuwa Taro
E-mail:taro@shuwasystem.ne.jp
-このページの感想を選んでください-
大変面白い
-その他なんでもコメントお願いいたします-
Javaっておもしろいですね
メールを送る　リセットする
ここにメッセージが表示されます…。

After

*メール送信時に挨拶を表示する
お名前:Shuwa Taro
E-mail:taro@shuwasystem.ne.jp
-このページの感想を選んでください-
大変面白い
-その他なんでもコメントお願いいたします-
Javaっておもしろいですね
メールを送る　リセットする
ありがとうございます!!
アンケートにご回答いただきありがとうございます(^o^)/。
あなたの貴重なご意見は今後のHomePage作りに役立てさせていただきます。
メールの内容は下の通りです。もし内容に不都合な点があれば
hamba@bekkoame.ne.jpまでご連絡願います。
お名前:Shuwa Taro
E-mail:taro@shuwasystem.ne.jp
感想:大変面白い
コメント:Javaっておもしろいですね

解説

サンプルでは、「メールを送る」ボタンが押されると、メールを送る動作と同時に、フレームにお礼とメールの内容を表示するようにしています。
CGIなどは、送信されたデータをサーバーが正常に受け取った後に、内容確認の文章を表示します。ところが、JavaScriptの場合、イベントハンドラ「onSubmit」は「submit」ボタンが押された時にJavaScriptの処理を実行しているので、送信が正常に行われたかどうかの判断ができない点に注意してください。
フォーム内の各値の書き出しは、textオブジェクトは「document.フォームオブジェクト名.textオブジェクト名.value」、optionオブジェクトは「document.フォームオブジェクト名.selectオブジェクト名.options[document.フォームオブジェクト名.selectオブジェクト名.selectedIndex].value」としています。各オブジェクトの階層関係には、十分に注意してください。

Sample

【フレーム】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<FRAMESET ROWS="*,200">
    <FRAME SRC="f1.html" name="f1">
    <FRAME SRC="f2.html" name="f2">
</FRAMESET>
```

```
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function Mopen(){
    parent.f2.document.open();
    parent.f2.document.write("<HTML>");
    parent.f2.document.write("<HEAD><TITLE>ArigatouGoza
imas</TITLE></HEAD>");
    parent.f2.document.write("<BODY BGCOLOR='#FFFFFF'>");
    parent.f2.document.write("<CENTER><B>*ありがとうございます!!
*</B></CENTER>");
    parent.f2.document.write("<BR>");
    parent.f2.document.write("<IMG SRC='image.gif' ALIG
N='left'>");
    parent.f2.document.write("アンケートにご回答いただきありがとうご
ざいます(^o^)/。");
    parent.f2.document.write("<BR>");
    parent.f2.document.write("あなたの貴重なご意見は今後のHomePa
ge作りに役立てさせていただきます。");
    parent.f2.document.write("<BR>");
    parent.f2.document.write("メールの内容は下の通りです。もし内容に
不都合な点があれば");
    parent.f2.document.write("<BR>");
    parent.f2.document.write("<A HREF='mailto:hamba@be
kkoame.ne.jp'>hamba@bekkoame.ne.jp</A>");
    parent.f2.document.write("までご連絡願います。");
    parent.f2.document.write("<HR>");
    parent.f2.document.write("お名前:".bold());
    parent.f2.document.write(document.MAIL1.Namae.value);
    parent.f2.document.write("<BR>");
    parent.f2.document.write("E-mail:".bold());
    parent.f2.document.write(document.MAIL1.EMail.value);
    parent.f2.document.write("<BR>");
    parent.f2.document.write("感想:".bold());
    parent.f2.document.write(document.MAIL1.Kansou.
options[document.MAIL1.Kansou.selectedIndex].value);
    parent.f2.document.write("<BR>");
    parent.f2.document.write("コメント:".bold());
    parent.f2.document.write(document.MAIL1.Comment.
value);
```

```
        parent.f2.document.write("</BODY>");
        parent.f2.document.write("</HTML>");
        parent.f2.document.close();
         }
//--->
</SCRIPT>
<HEAD>
<BODY BGCOLOR="#FFFFFF">
*メール送信時に挨拶を表示する<P>
<FORM NAME="MAIL1" ACTION="mailto:hamba@bekkoame.ne.jp"
METHOD="POST" onSubmit="Mopen()">
<B>お名前:</B><INPUT TYPE="text" NAME="Namae" SIZE=40><BR>
<B>E-mail:</B><INPUT TYPE="text" NAME="EMail" SIZE=45><BR>
<p>
<B>-このページの感想を選んでください-</B><BR>
<SELECT NAME="Kansou" >
    <OPTION VALUE="大変面白い">大変面白い
    <OPTION VALUE="面白いよ">面白いよ
    <OPTION VALUE="ふつうだな">ふつうだな
    <OPTION VALUE="つまんないよ">つまんないよ
    <OPTION VALUE="よくわからん!!">よくわからん!!
</SELECT>
<P>
<B>-その他なんでもコメントお願いいたします-</B><BR>
<TEXTAREA NAME="Comment" ROWS=3 COLS=50>
</TEXTAREA>
<P>
<INPUT TYPE="submit" NAME="BOOK1" VALUE="メールを送る">
<INPUT TYPE="reset" VALUE="リセットする"><BR>
</FORM>
</BODY>
</HTML>
```

【f2.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
ここにメッセージが表示されます...。
</BODY>
</HTML>
```

parent→「frameオブジェクト」の「入力されたURLを別フレームに表示する」:P.324参照
document.open()→「documentオブジェクト」の「開いたウィンドウに文字を記述する」:P.337参照
document.close()→documentオブジェクト」の「開いたウィンドウに文字を記述する」:P.337参照
bold()→「stringオブジェクト」の「太字(ボールド)にする」:P.479参照

JavaScript

パスワードを入力する

<INPUT TYPE="password" NAME="パスワードオブジェクト名" value="値" イベントハンドラ>

Before

*パスワードを入力する

パスワードは"mh123"。
入力したらフォーム以外の所をクリックしてください。

*****

After

正しいパスワードが入力されました

Microsoft Internet Explorer

入力された暗証番号1234は間違いです!!

OK

解説

passwordオブジェクトは、一見すると普通のテキストの入力欄と同じですが、このフォーム内に入力された値は、黒丸などの他からは判別できない文字に置き換えられて表示されます。
サンプルでは、パスワードを入力してもらい、それが正しければ別のページをロードし、間違っていれば警告を出します。JavaScriptは基本的にソースコードを隠すようにはできていませんので、パスワードを完全に隠匿することは難しく、passwordオブジェクトを実際に使う場面としては、CGIの入力用インターフェイスとして使用することを前提としていると思われます。
なお、passwordオブジェクトはJavaScript1.0からのJavaScriptですが、Netscape Navigator2.0では値を正確に取得できない場合があります。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function  GetP(s) {
    if (s=="mh123")  { location.href = "OK.html" }
     else { alert("入力された暗証番号"+s+"は間違いです!!") }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*パスワードを入力する<P>
パスワードは"mh123"。<BR>
入力したらフォーム以外の所をクリックしてください。<BR>
<FORM NAME="ANSHYO">
<INPUT TYPE="password" NAME="anshyo" value="" onBlur=
"GetP(this.value)">
</FORM>
</BODY>
</HTML>
```

<INPUT TYPE="password">→「入力フォーム」の「パスワードの入力フィールドを作る」:P.145参照
alert()→「windowオブジェクト」の「警告用のダイアログボックスを開く」:P.287参照
onBlur→リファレンス「イベントハンドラ」の「onBlur」:P.547参照

Tip

JavaScriptで「"」を書き出す方法

JavaScriptでは、文字列をダブルクォーテーションマーク「"」で囲んで指定する必要があり、その中で「"」を使うことができません。

では、JavaScriptでダブルクォーテーションマーク「"」を書き出したい時には、どうすれば良いのでしょうか?

まず1つの方法として、特殊キャラクター文字の「¥"」を使う方法があります。そしてもう1つは、サンプルのように、「"」と「'」のネスト(入れ子)の関係を逆にして、文字列をシングルクォーテーションマーク「'」で囲み、その中で「"」を使う、という方法があります。

JavaScript

リセットしてもいいか確認する

onReset **[イベントハンドラ]**

Before

*リセットしてもいいか確認する

お名前: ShuwaTarou

メールを送る　リセットする

After

Microsoft Internet Explorer

フォームの内容をリセットします。
よろしければ"Yes"(OK)を押してください。

OK　キャンセル

解説

イベントハンドラ「onReset」は、リセットボタンが押された時のイベントを取得します。サンプルでは、「リセットする」のボタンが押された時に確認のダイアログボックスを開き、Yes(OK)が押された時はフォームをリセットし、そうでない時はリセットの処理を中止しています。
JavaScript1.1で追加されたイベントハンドラです。

Sample

```
<HTML>
<HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
function Mcheck() {
    if ( confirm ('フォームの内容をリセットします。¥nよろしければ"Yes"
(OK)を押してください。')) { return true }
return false }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*リセットしてもいいか確認する<P>
<FORM NAME="MAIL4" ACTION="mailto:hamba@ppp.bekkoame.
or.jp" METHOD="POST" onReset="return Mcheck()">
<B>お名前:</B><INPUT NAME="Namae" SIZE=20><BR>
<P>
<INPUT TYPE="submit" NAME="BOOK1" VALUE="メールを送る">
<INPUT TYPE="reset" VALUE="リセットする">
</FORM>
</BODY>
</HTML>
```

confirm()→「windowオブジェクト」の「確認ボタン付きのダイアログボックスを開く」:P.288参照

アップロードするファイルを選ぶ

<INPUT TYPE="file" NAME="ファイルオブジェクト名">
document.フォーム名.ファイルオブジェクト名.value

Before / **After**

解説

FileUploadオブジェクトは、アップロードするファイルを選択するフォームです。「Browse...」ボタンをクリックするとローカルのディレクトリが参照でき、選択するとテキスト欄にファイル名が表示されます。その時に、「value」にはディレクトリの情報が納められ、「document.フォーム名.fileオブジェクト名.value」で参照することができます。
FileUploadオブジェクトは、あくまでもアップロードするファイルを選択することしかできず、実際にアップロードする作業は、HTMLやCGIの力を借りることになります。これは、JavaScriptがセキュリティ確保のため、ローカルのリソースにアクセスすることを厳しく禁止しているためです。
JavaScript1.1から追加されたオブジェクトです。

Sample

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*アップロードするファイルを選ぶ<P>
<FORM NAME="form1">
アップロードするファイル：<INPUT TYPE="file" NAME="UploadFile">
<P>
<INPUT TYPE="button" VALUE="Faile情報"
   onClick="alert('このファイルのディレクトリは' + document.form1.
UploadFile.value+'です。')">
</FORM>
</BODY>
</HTML>
```

<INPUT TYPE="file">→「入力フォーム」の「ファイルを選択する」：P.162参照
alert()→「windowオブジェクト」の「警告用のダイアログボックスを開く」：P.287参照

JavaScript

フォームの内容を後から変える

<INPUT TYPE="radio" NAME="*ラジオオブジェクト名*" VALUE="*値*" *イベントハンドラ*>
<SELECT NAME="*セレクトオブジェクト名*">
<OPTION>
options[*インデックス*] **[プロパティ]**

解説 JavaScript1.1から、フォームの内容を後から変更できるようになりました。
サンプルでは、「options[インデックス]」で、「男の子用」・「女の子用」の2種類のoptionオブジェクトの配列を作り、それをラジオボタンのクリックで切り替えています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
function BY(PR) {
    PR.pr.options[0].text = "男の子用のプレゼント";
    PR.pr.options[1].text = "プラモデル";
    PR.pr.options[2].text = "スニーカー";
    PR.pr.options[3].text = "おもちゃの缶詰(男の子用)";
}
function GR(PR) {
    PR.pr.options[0].text = "女の子用のプレゼント";
    PR.pr.options[1].text = "テディベアー";
    PR.pr.options[2].text = "ドレス";
    PR.pr.options[3].text = "おもちゃの缶詰(女の子用)";
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*フォームの内容を後から変える<P>
<HR>
男の子用・女の子用の 2 通りのプレゼントを用意しました。どちらかのラジオボタンをク
リックしてから、選んでください。<BR>
<FORM NAME="BORG">
 <INPUT TYPE="radio" NAME="borg" VALUE="BOY" onClick="BY
(this.form)">男の子用
<BR>
 <INPUT TYPE="radio" NAME="borg" VALUE="GIR" onClick="GR
(this.form)">女の子用
<P>
<SELECT NAME="pr">
<OPTION>男の子用・女の子用どちらか選んで下さい
<OPTION>------------------------------------
<OPTION>------------------------------------
<OPTION>------------------------------------
<OPTION>
</FORM>
</BODY>
</HTML>
```

<INPUT TYPE="radio">→「入力フォーム」の「ラジオボタンを作る」:P.155参照

JavaScript

フォームのタイプを調べる

document.フォームオブジェクト名.elements[*i*].type 【プロパティ】

*フォームのタイプを調べる

Button
CheckBox
RadioButton
SELECT
SubmitButton
TextBox
TextArea
****
ResetButton
隠しテキストボックス
参照...

1番目のフォームのタイプ: button
2番目のフォームのタイプ: checkbox
3番目のフォームのタイプ: radio
4番目のフォームのタイプ: select-one
5番目のフォームのタイプ: submit
6番目のフォームのタイプ: text
7番目のフォームのタイプ: textarea
8番目のフォームのタイプ: password
9番目のフォームのタイプ: reset
10番目のフォームのタイプ: hidden
11番目のフォームのタイプ: file

解説

「type」プロパティは、「button」や「checkbox」などのフォームのタイプを返すプロパティです。
サンプルでは、フォームオブジェクト「FTYPE」内の各フォーム関連のオブジェクトを配列として捉え、上から順番にタイプを書き出しています。
JavaScript1.1で追加されたプロパティです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*フォームのタイプを調べる<P>
<FORM NAME="FTYPE">
    <INPUT TYPE="button" NAME="BUTTON" value="Button"><BR>
    <INPUT TYPE="checkbox" NAME="CHECBOX" VALUE="Check
Box" checked>CheckBox<BR>
    <INPUT TYPE="radio" NAME="RADIO" value="RadioButton"
checked>RadioButton<BR>
    <SELECT NAME="SELECT">
        <OPTION>SELECT
        <OPTION>SELECT1
    </SELECT><BR>
    <INPUT TYPE="submit" NAME="SUBMIT" VALUE="SubmitBut
ton"><BR>
    <INPUT TYPE="text" NAME="TEXT" VALUE="TextBox" SIZE=
10 ><BR>
    <TEXTAREA NAME="TEXTAREA" ROWS=3 COLS=20 >TextArea
</TEXTAREA><BR>
    <INPUT TYPE="password" VALUE="Password" NAME="PASSWOR
D" VALUE="Password" SIZE=15><BR>
    <INPUT TYPE="reset" NAME="RESET" VALUE="ResetButton
"><BR>
    <INPUT TYPE="hidden" NAME="HIDDEN" VALUE="Hidden">
隠しテキストボックス<BR>
    <INPUT TYPE="file" NAME="UploadFile">
</FORM>
<HR>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
var L = document.FTYPE.elements.length
for(i=0; i<L; i++){
document.write( i+1 + "番目のフォームのタイプ:".bold())
document.write(document.FTYPE.elements[i].type)
document.write("<BR>")
}
//--->
</SCRIPT>
</BODY>
</HTML>
```

bold()→「stringオブジェクト」の「太字(ボールド)にする」:P.479参照

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

マップエリア外のクリックで警告ウィンドウを開く

JavaScript:関数

Before

*マップエリア外のクリックで警告ウィンドウを開く

After

Microsoft Internet Explorer

イメージマップのエリア外です!!

OK

解説

サンプルでは、MAPのリンク指定したい部分がクリックされた時は「JavaScript:」で関数を発生させてページを読み込み、範囲外をクリックされた時は警告用のダイアログボックスを開くようにしています。
このサンプルスクリプトはNetscape Navigator2.xでも使用できます。

Sample

【メイン】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function Go(url){ location.href=url }
function Miss() { alert("イメージマップのエリア外です!!") }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*マップエリア外のクリックで警告ウィンドウを開く<P>
<MAP NAME="ARIA1">
    <AREA NAME="aria1" SHAPE=rect COORDS=13,9,73,69 HREF
="JavaScript:Go('MAP1.html')">
    <AREA NAME="aria2" SHAPE=circle COORDS=119,38,29 HR
EF="JavaScript:Go('MAP2.html')">
    <AREA NAME="aria3" SHAPE=poly COORDS=160,69,188,7,22
2,69 HREF="JavaScript:Go('MAP3.html')">
    <AREA NAME="aria4" SHAPE=rect COORDS=0,0,234,81 HREF
="JavaScript:Miss()">
</MAP>
```

```
<IMG SRC="MAP.gif" USEMAP="#ARIA1" BORDER=0 WIDTH="234" HEIGHT="81" ALT="ARIATEST" >
</BODY>
</HTML>
```

【MAP1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<IMG SRC="MAP1.gif" WIDTH="60" HEIGHT="59" ALT="Midori Sikaku"><HR>
<B>*緑の四角*</B>
</BODY>
</HTML>
```

同様にして"MAP1.html～MAP3.html"を用意します。

<MAP>･<AREA>→「画像とマルチメディア」の「イメージマップを作成する」:P.132参照

alert→「windowオブジェクト」の「警告用のダイアログボックスを開く」:P.287参照

focus()→「windowオブジェクト」の「ウィンドウを前に出す」:P.304参照

location.href→「locationオブジェクト」の「自ページURLのを取得する」:P.353参照

Tip

エリアマップ内でNetscape Navigator2.0でも使えるJavaScriptは?

Areaオブジェクトは、JavaScript1.1から追加されたオブジェクトですが、<AREA>タグ内で「HREF="JavaScript:関数"」を使えば、Netscape Navigator2.0でも機能するスクリプトを作ることができます。これは、AreaオブジェクトがLinkの配列内にあるためだと思われます。

つまり、Netscape Navigator2.0では、<AREA>タグ内の「HREF="JavaScript:関数"」は、リンクオブジェクトとして捉えられているのでしょう。

しかし、<AREA>タグに記された「onMouseOver」や「onMouseOut」などのイベントハンドラは、Netscape Navigator2.0(JavaScript1.0)からサポートされているはずのものに関しても、Netscape Navigator2.0では評価されません。

したがって、例えば、サンプル「フォームに説明を出す」(次ページ)や「イメージマップをリンク以外の機能で使う」(P.398)は、Netscape Navigator2.0で実行した場合、フォームに文字を書き出すスクリプトの部分は動きませんが、その他の、新しいウィンドウを開いたり、バックの色を変える部分のスクリプトは正常に動きます。

フォームに説明を出す

HREF="javascript:関数"
onMouseOver 【イベントハンドラ】
onMouseOut 【イベントハンドラ】

*フォームに説明を出す
緑の四角

*フォームに説明を出す
黄色の丸

*フォームに説明を出す
赤の三角

解説

サンプルでは、指定されたエリア内にマウスカーソルが乗った時に、「onMouseOver」がフォーム内に文字を書き出します。エリア内からマウスカーソルが離れた時は、「onMouseOut」が関数「MessCr()」を発生させてフォームに「""」の値を引き渡し、それによってフォーム内の文字を消去しています。
「onMouseOver」は、Netscape Navigator2.0から使用可能なイベントハンドラですが、<AREA>タグ内では評価されません。
なお、リンクをクリックすると、新しいページが開きます。

Sample

【メイン】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
function Go(url) { window.open( url ,"IWindow" ) }
function Mis() { alert("イメージマップのエリア外です!!") }
function MessCr() { document.Fmess.fmess.value = "" }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*フォームに説明を出す<P>
<MAP NAME="ARIA1">
    <AREA SHAPE=rect COORDS=13,9,73,69 HREF="JavaScript:
Go('MAP1.html')"
```

```
        onMouseOver="document.Fmess.fmess.value = '緑の四角'"
        onMouseOut="MessCr()">
        <AREA SHAPE=circle COORDS=119,38,29 HREF="JavaScript:
Go('MAP2.html')"
        onMouseOver="document.Fmess.fmess.value = '黄色の丸'"
        onMouseOut="MessCr()">
        <AREA SHAPE=poly COORDS=160,69,188,7,222,69 HREF="
JavaScript:Go('MAP3.html')"
        onMouseOver="document.Fmess.fmess.value = '赤の三角'"
        onMouseOut="MessCr()">
        <AREA SHAPE=rect COORDS=0,0,234,81 HREF="JavaScript:
Mis()"
         onMouseOver="MessCr()">
</MAP>
<IMG SRC="MAP.gif" USEMAP="#ARIA1" BORDER=0 WIDTH="234"
HEIGHT="81" ALT="ARIATEST" >
<FORM NAME="Fmess">
<INPUT TYPE="text" NAME="fmess" SIZE=20>
</FORM>
</BODY>
</HTML>
```

【MAP1.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
focus()
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<IMG SRC="MAP1.gif" WIDTH="60" HEIGHT="59" ALT="Midori
Sikaku"><HR>
<B>*緑の四角*</B>
</BODY>
</HTML>
```

同様にして"MAP1.html〜MAP3.html"を用意します。

<MAP>→「画像とマルチメディア」の「イメージマップを作成する」:P.132参照

alert→「windowオブジェクト」の「警告用のダイアログボックスを開く」:P.287参照

window.open→「windowオブジェクト」の「新しいウィンドウを開く」:P.296参照

focus()→「windowオブジェクト」の「ウィンドウを前に出す」:P.304参照

location.href→「locationオブジェクト」の「自ページURLのを取得する」:P.353参照

<FORM>→「formオブジェクト」の「フォームに文字を流す」:P.376参照

イメージマップをリンク以外の機能で使う

JavaScript

HREF="javascript:*関数*"
onMouseOver 「イベントハンドラ」
onMouseOut 「イベントハンドラ」

*イメージマップをリンク以外の機能で使う
バックを緑に

*イメージマップをリンク以外の機能で使う
バックを緑に

*イメージマップをリンク以外の機能で使う
バックを黄色に

*イメージマップをリンク以外の機能で使う
バックを赤に

解説 サンプルでは、指定されたエリアをクリックすると、「JavaScript:」が関数「BdColor()」を発生させて、中で指定している色の値を「document.bgColor」に引き渡し、バックの色を変更しています。

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function BdColor(BC) { document.bgColor=BC }
function Mis() { alert("イメージマップのエリア外です!!") }
function MessCr() { document.Fmess.fmess.value = "" }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*イメージマップをリンク以外の機能で使う<P>
<MAP NAME="ARIA1">
    <AREA NAME="aria1" SHAPE=rect COORDS=13,9,73,69 HREF
="JavaScript:BdColor('green')"
     onMouseOver="document.Fmess.fmess.value = 'バックを緑に'"
     onMouseOut="MessCr()">
     <AREA NAME="aria2" SHAPE=circle COORDS=119,38,29
HREF="JavaScript:BdColor('yellow')"
     onMouseOver="document.Fmess.fmess.value = 'バックを黄色に'"
     onMouseOut="MessCr()" >
    <AREA NAME="aria3" SHAPE=poly COORDS=160,69,188,7,22
2,69 HREF="JavaScript:BdColor('red')"
     onMouseOver="document.Fmess.fmess.value = 'バックを赤に'"
     onMouseOut="MessCr()">
    <AREA NAME="aria4" SHAPE=rect COORDS=0,0,234,81 HREF
="JavaScript:Mis()"
     onMouseOver="MessCr()">
</MAP>
<IMG SRC="MAP.gif" USEMAP="#ARIA1" BORDER=0 WIDTH="234"
HEIGHT="81" ALT="ARIATEST" >
<FORM NAME="Fmess">
<INPUT TYPE="text" NAME="fmess" SIZE=20>
</FORM>
</BODY>
</HTML>
```


<MAP>→「画像とマルチメディア」の「イメージマップを作成する」:P.132参照

alert→「windowオブジェクト」の「警告用のダイアログボックスを開く」:P.287参照

document.bgColor→「documentオブジェクト」の「背景色を変えるボタンを作る」:P.344参照

<INPUT TYPE="text">→「formオブジェクト」の「フォームに文字を流す」:P.376参照

画像の情報を取得する

document.オブジェクト名.border	[プロパティ]
document.オブジェクト名.complete	[プロパティ]
document.オブジェクト名.height	[プロパティ]
document.オブジェクト名.hspace	[プロパティ]
document.オブジェクト名.lowsrc	[プロパティ]
document.images[インデックス].src	[プロパティ]
document.images[インデックス].vspace	[プロパティ]
document.images[インデックス].width	[プロパティ]

*画像の情報を取得する

ボーダー:2
ロードが終わったか:true
イメージの高さ:100
イメージのhspace:2
lowsrcのURL:image1.gif
イメージのURL:file://Pro_server/souko/urata/2-ナビオブジェ-ToU/12_01/image2.gif
イメージのvspace:2
イメージの幅:100

解説

Imageオブジェクトは、ページ上の画像の[0]から始まる配列を作成します。Imageオブジェクトの情報は、<IMG>タグ内で設定した「NAME」で参照する以外に、配列でも参照できます。

「border」プロパティはボーダーの値を、「complete」プロパティは画像のロードが終っていれば「true」、終っていなければ「false」の値を、「height」プロパティは画像の高さの値を、「hspace」プロパティはドキュメントとの横方向の間隔の値を、「lowsrc」プロパティは、正式な画像を表示する前に変わりに表示する、低解像度の画像のURLを、「src」プロパティは画像ファイルのURLを、「vspace」プロパティはドキュメントとの立て方向の間隔の値を、「width」プロパティは画像の幅の値を、それぞれ返します。

「src」プロパティ以外は、読み込み専用で、後から変更することはできません。しかし、「src」プロパティの値は変更可能なので、それを替えることによってイメージを置き換えることができます。

JavaScript1.1で追加されたオブジェクトです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*画像の情報を取得する<P>
<IMG SRC="image2.gif" NAME="IMG" ALT="image.gif" LOWSRC
="image1.gif" BORDER=2 HSPACE="2" VSPACE="2">
<P>
<SCRIPT Language="JavaScript1.1">
<!---
document.write("ボーダー:");
document.write(document.IMG.border);
document.write("<BR>");
document.write("ロードが終わったか:");
document.write(document.IMG.complete);
document.write("<BR>");
document.write("イメージの高さ:");
document.write(document.IMG.height);
document.write("<BR>");
document.write("イメージのhspace:");
document.write(document.IMG.hspace);
document.write("<BR>");
document.write("lowsrcのURL:");
document.write(document.IMG.lowsrc);
document.write("<BR>");
document.write("イメージのURL:");
document.write(document.images[0].src);
document.write("<BR>");
document.write("イメージのvspace:");
document.write(document.images[0].vspace);
document.write("<BR>");
document.write("イメージの幅:");
document.write(document.images[0].width);
//--->
</SCRIPT>
</BODY>
</HTML>
```

<IMG>タグ→「画像とマルチメディア」の「画像を配置する」:P.123参照

JavaScript

画像をアニメーションする

```
オブジェクト名 = new Image()
document.オブジェクト名.src
```

[プロパティ]

解説

「src」プロパティが後から変更できることを利用して、複数の画像をリアルタイムに切り替えて、アニメーションのような効果を出すことができます。

サンプルでは、まずimage1.gifからimage5.gifまでの5枚の画像を用意し、Arrayオブジェクトを使ってImageオブジェクトの配列を作成しています。配列内の要素には、「ANIMA[1]」から「ANIMA[5]」の5つのImageオブジェクトがあり、それぞれ「ANIMA[1].src」に「image1.gif」の値、「ANIMA[2].src」に「image2.gif」の値、といった具合に画像ファイルのURLの値を設定しています。

画像ファイル「animation」が読み込まれた時に、「setTimeout()」を使って500ミリ秒ごとに、「ANIMA[1].src」〜「ANIMA[5].src」を順番に繰り返して「animation」に置き換えることによって、画像をアニメーションさせています。

「setTimeout()」は、指定された時間後に1度だけ処理を行うメソッドですが、画像が置き換わる度に呼び出されるので、結果的にアニメーションの処理は、終ることなく繰り返されます。

JavaScript1.1で追加されたオブジェクトです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT Language="JavaScript1.1">
<!---
var ImageSetB = 1 ;
ANIMA = new Array()
for(i = 1; i < 6; i++) {
   ANIMA[i] = new Image() ;
   ANIMA[i].src = "image" + i + ".gif" ;
}
function anime_1() {
   document.animation.src = ANIMA[ImageSetB].src ;
   ImageSetB++ ;
       if(ImageSetB > 5) {
       ImageSetB = 1 ;
       }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*画像をアニメーションする<P>
<IMG SRC="image1.gif" NAME="animation" ALT="Animation"
 BORDER=0 WIDTH="100" HEIGHT="100"
 onLoad="setTimeout('anime_1()',500)">
</BODY>
</HTML>
```

setTimeout()→「windowオブジェクト」の「ステータス行に文字を流す」:P.292参照

オブジェクト = new Array()→「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う -」:P.494参照

NN3.0 NN4.0 NN4.06 IE4.0 IE5.0

JavaScript

イメージを操作する

JavaScript アニメーションにスタート・ストップボタンを付ける

```
オブジェクト名 = new Image()
document.オブジェクト名.src
```

[プロパティ]

*アニメーションにスタート・ストップボタンを付ける

スタート ストップ

解説

サンプルでは、「画像をアニメーションする」(前項)で作成したImageブジェクトの配列を利用しています。
「スタート」ボタンが押された時は、関数「anime_2()」が発生してImageオブジェクトの配列の要素が呼び出され、アニメーションがスタートします。「ストップ」ボタンが押された時には、関数「stop()」が発生し、「clearTimeout()」によってアニメーションが停止します。
「clearTimeout()」の設定は、「ID名=setTimeout()」とsetTimeout()にIDを設定し、そのIDを「clearTimeout(ID名)」とclearTimeout()内で指定することにより行います。
JavaScript1.1で追加されたオブジェクトです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT Language="JavaScript1.1">
<!---
var TimeSet1 = 500 ;
var ImageSetA = 1 ;
ANIMA = new Array() ;
for(i = 1; i < 6; i++) {
   ANIMA[i] = new Image() ;
   ANIMA[i].src = "image" + i + ".gif" ;
}
function anime_2() {
   document.animation.src = ANIMA[ImageSetA].src ;
   ImageSetA++ ;
       if( ImageSetA > 5) {
       ImageSetA = 1 ;
       }
   timerID=setTimeout("anime_2()", TimeSet1) ;
}
function stop(){
   clearTimeout(timerID) ;
}
//---->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*アニメーションにスタート・ストップボタンを付ける<P>
<IMG SRC="image1.gif" NAME="animation" ALT="Animation"
 BORDER=0 WIDTH="100" HEIGHT="100">
<FORM>
   <INPUT TYPE="button" VALUE="スタート" onClick="anime_2()">
   <INPUT TYPE="button" VALUE="ストップ" onClick="stop()">
</FORM>
</BODY>
</HTML>
```


setTimeout()→「windowオブジェクト」の「ステータス行に文字を流す」:P.292参照

clearTimeout()→「windowオブジェクト」の「ステータス行に文字を流す」:P.292参照

オブジェクト = new Array()→「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う - 」:P.494参照

NN3.0
NN4.0
NN4.06
IE4.0
IE5.0

JavaScript

画像に触ったりクリックした時に画像を変化させる

```
オブジェクト名 = new Image()
document.オブジェクト名.src          [プロパティ]
document.images[インデックス]        [配列]
```

*画像に触ったりクリックした時に画像を変化させる

*画像に触ったりクリックした時に画像を変化させる

*画像に触ったりクリックした時に画像を変化させる

解説

サンプルではまず、普通の時のボタンの画像「button1.gif」とマウスが画像の上に乗った時の画像「button2.gif」、ボタンがクリックされた時の画像「button3.gif」、の3枚の画像を用意し、「画像をアニメーションする」(P.402)と同じ要領で、それぞれのURLの値を持った3つの配列の要素を作っています。

実際に画像を変化させるのは、リンクの中に設定したイベントハンドラで行います。どの画像を変化させるかは、HTMLファイルが読み込まれると同時に、「document.images[0]」から始まるイメージ配列ができているので、それで指定するようにしています。

上の画像上にマウスが乗った時は、イベントハンドラ「onMouseOver」が関数「SetImage1(2,0)」を発生して、「document.images[0]」の位置の画像を「button2.gif」に置き換えます。クリックした時は、「onClick="SetImage1(3,0)"」によって、「document.images[0]」の位置のイメージを「button3.gif」に置き換えます。マウスが画像から離れた時は、「onMouseOut="SetImage1(1,0)"」によって、「document.images[0]」の位置の画像を、「button1.gif」に置き換えています。

また、下の画像上にマウスが乗ったり、クリックされたり、マウスが離れた時は、「document.images[1]」の位置の画像が置き換わるように設定しています。

JavaScript1.1で追加されたオブジェクトです。

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT Language="JavaScript1.1">
<!---
var ButtonImage = new Array() ;
    for(i = 1; i < 4; i++) {
        ButtonImage[i]= new Image() ;
        ButtonImage[i].src="button" + i + ".gif" ;
    }
function SetImage1(flag, position) {
        document.images[position].src=ButtonImage[flag].
src ;
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*画像に触ったりクリックした時に画像を変化させる<P>
<A HREF="#" onMouseOver="SetImage1(2,0)" onMouseOut="Set
Image1(1,0)" onClick="SetImage1(3,0)">
<IMG SRC="button1.gif" ALT="button1" BORDER=0 WIDTH="78"
HEIGHT="33"></A>
<P>
<A HREF="#" onMouseOver="SetImage1(2,1)" onMouseOut="Set
Image1(1,1)" onClick="SetImage1(3,1)">
<IMG SRC="button1.gif" ALT="button2"  BORDER=0 WIDTH=
"78" HEIGHT="33"></A>
</BODY>
</HTML>
```

オブジェクト = new Array()→「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う -」:P.494参照

重要 マウス操作のタイミングで画像を置き換える時の注意

Imageオブジェクトでは、イベントハンドラ「onClick」・「onMouseOver」・「onMouseOut」はサポートされていません。

したがって、マウスが画像上に乗った時や離れた時などに画像置き換え等の処理を行う場合は、このサンプルのように、Linkオブジェクト内などにイベントハンドラを置く必要があります。

もし、画像をクリックした時に違うページへ飛びたくない場合は、「画像に触った時に別の画像を変化させる」(次項)のように、"return false"を返してイベントの処理を中止させるか、「HREF="javascript:"」を使えば大丈夫です。

JavaScript

画像に触った時に別の画像を変化させる

```
オブジェクト名 = new Image()
document.オブジェクト名.src          [プロパティ]
document.images[インデックス]          [配列]
```

*画像に触った時に別の画像を変化させる

*画像に触った時に別の画像を変化させる

*画像に触った時に別の画像を変化させる

*画像に触った時に別の画像を変化させる

解説

「画像に触ったりクリックした時に画像を変化させる」(前項)と同じ要領で、画像に触ったタイミングで別の画像を置き換えることができます。
サンプルでは、イメージ配列"document.images[0]"の画像の上にマウスが乗った時に、"document.images[1]"の画像が置き換わるようにしています。
JavaScript1.1で追加されたオブジェクトです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT Language="JavaScript1.1">
<!---
OnMouse = new Array() ;
    for(i = 1; i < 5; i++) {
        OnMouse[i] = new Image() ;
        OnMouse[i].src = "image" + i + ".gif" ;
    }
function OnMoSet1(flag, position) {
    document.images[position].src=OnMouse[flag].src ;
    return false ;
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*画像に触った時に別の画像を変化させる<P>
<A HREF="#" onMouseOver="OnMoSet1(2,1)" onMouseOut="OnMo
Set1(1,1)" onClick="return OnMoSet1(4,1)">
<IMG SRC="image1.gif" ALT="OnMouseA-1" BORDER=0 WIDTH="
100" HEIGHT="100"></A>
<A HREF="#" onMouseOver="OnMoSet1(3,0)" onMouseOut="OnMo
Set1(1,0)" onClick="return OnMoSet1(4,0)">
<IMG SRC="image1.gif" ALT="OnMouseA-2" BORDER=0 WIDTH="
100" HEIGHT="100"></A>
</BODY>
</HTML>
```

オブジェクト = new Array()→「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う -」:P.494参照

JavaScript

別フレームの画像を変化させる

```
オブジェクト名 = new Image()
document.オブジェクト名.src                    [プロパティ]
document.images[インデックス]                  [配列]
```

解説

「画像に触ったりクリックした時に画像を変化させる」(P.406)と同じ要領で、別フレームの画像を置き換えることができます。
その場合は、置き換える画像の指定は、「parent.画像のあるフレーム名.document.images[インデックス].src」とします。
JavaScript1.1で追加されたオブジェクトです。

Sample

【フレーム】

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<FRAMESET COLS="140,*">
    <FRAMESET ROWS="140,*">
        <FRAME SRC="f1.html" NAME="f1">
        <FRAME SRC="f2.html" NAME="f2">
    </FRAMESET>
<FRAME SRC="f3.html" NAME="f3">
</FRAMESET>
<NOFRAMES>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</NOFRAMES>
</HTML>
```

【f1.html】

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
<IMG SRC="image1.gif" ALT="OnMouseB-1" WIDTH="100" HEIGHT="100">
</BODY></HTML>
```

【f2.html】

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT Language="JavaScript1.1">
<!---
OnMouseB = new Array()
for(i = 1; i < 6; i++) {
   OnMouseB[i] = new Image() ;
   OnMouseB[i].src = "image" + i + ".gif" ;
}
function OnMoSet2(flag, position) {
    parent.f1.document.images[position].src=OnMouseB[flag].src ;
}
//--->
</SCRIPT>
<BODY BGCOLOR="#FFFFFF">
·<A HREF="fp1.html" TARGET="f3" onMouseOver="OnMoSet2(2,0)" onMouseOut="OnMoSet2(1,0)">笑顔</A><P>
·<A HREF="fp2.html" TARGET="f3" onMouseOver="OnMoSet2(3,0)" onMouseOut="OnMoSet2(1,0)">泣き顔</A><P>
·<A HREF="fp3.html" TARGET="f3" onMouseOver="OnMoSet2(4,0)" onMouseOut="OnMoSet2(1,0)">怒り顔</A><P>
·<A HREF="f3.html" TARGET="f3" onMouseOver="OnMoSet2(5,0)" onMouseOut="OnMoSet2(1,0)">始めへ戻る</A><P>
</BODY>
</HTML>
```

【f3.html】

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*別フレームの画像を変化させる
</BODY></HTML>
```


「フレーム」の「フレームの全体構成を指定する」:P.168参照

parent.フレーム名→「frameオブジェクト」の「入力されたURLを別フレームに表示する」:P.324参照

オブジェクト = new Array()→「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う -」:P.494参照

JavaScript

画像のロード状態を表示する

onAbort [イベントハンドラ]
onError [イベントハンドラ]
onLoad [イベントハンドラ]

*画像のロード状態を表示する
イメージをロードしています....

*画像のロード状態を表示する
イメージをロードしています....

*画像のロード状態を表示する
イメージのロードが終了しました

解説

イベントハンドラ「onAbort」は、画像読み込み中に中止ボタンが押されるなどして画像の読み込みが中止された時に、「onError」は画像読み込みエラー時に、「onLoad」は画像読み込み終了時に、それぞれイベントを発生します。
サンプルでは、画像ファイルのそれぞれの状態を取得し、それに合わせたメッセージをフォーム内に書き出しています。
このサンプルで、画像の後にフォームを持って来るようにすると、スクリプトが機能しない場合があります。
JavaScript1.1で追加されたオブジェクトです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT Language="JavaScript1.1">
<!---
function STOP(){ document.ZYOUTAI.zyo.value = "イメージの
ロードが中止されました" }
function ERR(){ document.ZYOUTAI.zyo.value = "イメージのロー
ドに失敗しました" }
```

```
function OK(){ document.ZYOUTAI.zyo.value = "イメージのロード
が終了しました" }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*画像のロード状態を表示する<P>
<FORM NAME="ZYOUTAI">
<INPUT TYPE="text" NAME ="zyo" VALUE = " イメージをロードしてい
ます...." SIZE=60 >
</FORM>
<BR>
<IMG SRC="usagi.gif" ALT="usagi.gif" WIDTH="408" HEIGHT
="300" onAbort="STOP()" onError="ERR()" onLoad="OK()">
<P>
</BODY>
</HTML>
```

document.フォーム名.エレメント名.value→「「formオブジェクト」の「メール送信時に挨拶を表示する」:P.383参照

onAbort→リファレンス「イベントハンドラ」の「onAbort」:P.547参照

onError→リファレンス「イベントハンドラ」の「onError」:P.548参照

onLoad→リファレンス「イベントハンドラ」の「onLoad」:P.548参照

注意

大きな画像やフォーム、テーブルとJavaScriptを共存させる時の注意

大きな画像やフォーム・テーブルなど、ブラウザがレイアウトを完成するのに時間がかかるようなHTMLの記述の後にJavaScriptを記述しても、スクリプトがうまく動かない場合があります。

これは、ブラウザのレイアウト確定に邪魔され、JavaScriptの認識が行なわれなかったために起こる現象のようです。

画像の場合、Netscape Navigator3.0以降のブラウザを使ったり、そうでなくても「WIDTH」と「HEIGHT」の指定をすれば、画像部分のレイアウト確定が早く行われるので、この問題はほぼ解消されます。けれども、テーブル内にJavaScriptを記述した時や、このサンプルのように大きな画像の上ではなく、下にフォームと組み合わせたJavaScriptを組みこんだ時などに、問題が起こる場合があります。

これらの問題は、Netscape Navigator3.0以降のブラウザでも発生する場合がありますが、Internet Explorerでこのような現象を確認したことはありません。

JavaScript

画像をリロードするか確認する

onError [イベントハンドラ]

Before

*画像をリロードするか確認する

このPageのイメージはリンクが切れています。

error.gif

After

Microsoft Internet Explorer

イメージのロードに失敗しました。ページをリロードしますか?

OK　キャンセル

解説

サンプルでは、画像の読み込みエラー時にイベントを発生するイベントハンドラ「onError」を利用して、画像が正常に読み込まれなかった時に、ページをリロードするかどうか確認するダイアログボックスを開いています。
JavaScript1.1で追加されたオブジェクトです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT Language="JavaScript1.1">
<!----
function ERR2(){
    if ( confirm ("イメージのロードに失敗しました。ページをリロードします
か?") ) { location.href="08ima.html" }
}
//--->
```

```
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*画像をリロードするか確認する<P>
このPageのイメージはリンクが切れています。<P>
<IMG SRC="error.gif" ALT="error.gif" WIDTH="474" HEIGHT
="198" onError="ERR2()">
</BODY>
</HTML>
```

---

confirm ()→「windowオブジェクト」の「確認ボタン付きのダイアログボックスを開く」:P.288参照
location.href→「locationオブジェクト」の「自ページのURLを取得する」:P.353参照
onError→リファレンス「イベントハンドラ」の「onError」:P.548参照

JavaScript

レイヤーの情報を取得する

【オブジェクトの作成】

```
document.layers[レイヤー名]
document.layers.レイヤー名
document.レイヤー名
document.layers[添番]

document.layers[レイヤー名].name          [プロパティ]
document.layers[レイヤー名].left          [プロパティ]
document.layers[レイヤー名].top           [プロパティ]
document.layers[レイヤー名].pageX         [プロパティ]
document.layers[レイヤー名].pageY         [プロパティ]
document.layers[レイヤー名].zIndex        [プロパティ]
document.layers[レイヤー名].visibility    [プロパティ]
```

*レイヤーの情報を取得する

オブジェクト名:LAY1
画面左からの位置:200
画面上からの位置:50
画面左からの位置:200
画面上からの位置:50
Z-Indexの値:1
レイヤーの可視属性:inherit

LAY1

解説

layerオブジェクトはdocumentオブジェクトのプロパティなので、オブジェクトの参照はサンプルのように「document.layers[レイヤー名]」・「document.layers.レイヤー名」・「document.レイヤー名」、または、オブジェクトを配列として取り扱って「document.layers[添番]」とすることにより可能です。

「name」プロパティは<LAYER>タグ内の「NAME」属性で設定したlayerオブジェクト名を、「left」プロパティはページあるいは親レイヤーの左上角からのレイヤー左上角のX軸上の位置の値を、「top」プロパティはページあるいは親レイヤーの左上角からのレイヤー左上角のX軸上の位置の値を、「pageX」プロパティはページ左上角からのレイヤー左上角のX軸上の位置の値を、「pageY」プロパティはページ左上角からのレイヤー左上角のY軸上の位置の値を、「zIndex」プロパティはレイヤーの軸上の値を、「visibility」プロパティはレイヤーの可視属性の値を、それぞれ返します。

親レイヤーの場合はページ左上角からの位置を参照するので、「left」プロパティと「pageX」プロパティ、「top」プロパティと「pageY」プロパティの値は同じになります。
また、サンプルでは、レイヤーに可視属性を設定していないので、親レイヤーの可視属性に準ずる「inherit」の値が返ります。
これらのプロパティは変更可能なので、ページ表示後にこれらの値を変更することにより、レイヤーを移動したり、可視属性を変化させることが可能です。

NN4.0
NN4.06

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*レイヤーの情報を取得する<P>
<LAYER NAME="LAY1" LEFT="200" TOP="50" WIDTH="200" HEIGHT
="150" Z-INDEX="1" BGCOLOR="#99FFCC">
<B>LAY1</B>
</LAYER>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("オブジェクト名:",document.layers["LAY1"].name);
document.write("<BR>");
document.write("画面左からの位置:",document.layers.LAY1.left);
document.write("<BR>");
document.write("画面上からの位置:",document.LAY1.top);
document.write("<BR>");
document.write("画面左からの位置:",document.layers[0].pageX);
document.write("<BR>");
document.write(" 画面上からの位置:",document.layers["LAY1"].
pageY);
document.write("<BR>");
document.write("Z-Index の値:",document.layers["LAY1"].z
Index);
document.write("<BR>");
document.write("レイヤーの可視属性:",document.layers["LAY1"].
visibility);
//-->
</SCRIPT>
</BODY>
</HTML>
```

<LAYER>→コラム「Netscape Navigator4.xのレイヤーについて・1～4」:P.204参照

子レイヤーの情報を取得する

【オブジェクトの作成】

```
document.layers[レイヤー名].layers[レイヤー名]
document.layers.レイヤー名.layers.レイヤー名
document.layers[添番].layers[添番]

document.layers[親レイヤー名].layers[子レイヤー名].name     [プロパティ]
document.layers[親レイヤー名].layers[子レイヤー名].left     [プロパティ]
document.layers[親レイヤー名].layers[子レイヤー名].top      [プロパティ]
document.layers[親レイヤー名].layers[子レイヤー名].pageX    [プロパティ]
document.layers[親レイヤー名].layers[子レイヤー名].pageY    [プロパティ]
document.layers[親レイヤー名].layers[子レイヤー名].zIndex   [プロパティ]
document.layers[親レイヤー名].layers
[子レイヤー名].visibility                                  [プロパティ]
```

*子レイヤーの情報を取得する

オブジェクト名:LAY2
親レイヤー左からの位置:50
親レイヤー上からの位置:50
画面左からの位置:250
画面上からの位置:100
Z-Indexの値:1
レイヤーの可視属性:show

子レイヤー

解説

ネストされた子レイヤーの参照は、サンプルのように「document.layers[親レイヤー名].layers[子レイヤー名]」・「document.layers.レイヤー名.layers.レイヤー名」、または「document.layers[添番].layers[添番]」とすることによって可能です。

子レイヤーの場合、「left」プロパティと「top」プロパティは親レイヤー左上角を基準とした、「pageX」プロパティと「pageY」プロパティはページ左上角を基準とした、レイヤーの位置の値を返します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*子レイヤーの情報を取得する<P>
<LAYER NAME="LAY1" LEFT="200" TOP="50" WIDTH="200" HEIGHT
="150" Z-INDEX="1" BGCOLOR="#99FFCC">
    <LAYER NAME="LAY2" LEFT="50" TOP="50" WIDTH="100" HE
IGHT="50" Z-INDEX="1" VISIBILITY="show" BGCOLOR="#FFFFCC">
<B>子レイヤー</B>
    </LAYER>
</LAYER>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("オブジェクト名:",document.layers["LAY1"].
layers["LAY2"].name);
document.write("<BR>");
document.write("親レイヤー左からの位置:",document.layers.LAY1.
layers.LAY2.left);
document.write("<BR>");
document.write("親レイヤー上からの位置:",document.layers[0].
layers[0].top);
document.write("<BR>");
document.write("画面左からの位置:",document.layers["LAY1"].
layers["LAY2"].pageX);
document.write("<BR>");
document.write("画面上からの位置:",document.layers["LAY1"].
layers["LAY2"].pageY);
document.write("<BR>");
document.write("Z-Index の値:",document.layers["LAY1"].
layers["LAY2"].zIndex);
document.write("<BR>");
document.write("レイヤーの可視属性:",document.layers["LAY1"].
layers["LAY2"].visibility);
//-->
</SCRIPT>
</BODY>
</HTML>
```

<LAYER>→コラム「Netscape Navigator4.xのレイヤーについて・1〜4」:P.204参照

クリップの情報を取得する

document.layers[レイヤー名].clip.top [プロパティ]
document.layers[レイヤー名].clip.left [プロパティ]
document.layers[レイヤー名].clip.right [プロパティ]
document.layers[レイヤー名].clip.bottom [プロパティ]
document.layers[レイヤー名].clip.width [プロパティ]
document.layers[レイヤー名].clip.height [プロパティ]

*クリップの情報を取得する

clip.top:20
clip.left:10
clip.right:40
clip.bottom:80
clip.wide:30
clip.heght:60

解説

レイヤーのクリップの値は、サンプルの通り「document.layers[レイヤー名].clip」で取得できます。
「clip.top」プロパティはレイヤー内のクリップの上の位置の値を、「clip.left」プロパティはレイヤー内のクリップ左側の位置の値を、「clip.right」プロパティはレイヤー内のクリップの右側の位置の値を、「clip.bottom」プロパティはレイヤー内のクリップの下の位置の値を、「clip.width」プロパティはクリップの幅の値を、「clip.height」プロパティはクリップの高さの値を、それぞれ返します。
これらのプロパティは、ページ表示後でも変更可能です。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*クリップの情報を取得する<P>
<LAYER NAME="LAY1" LEFT="200" TOP="50" WIDTH="200" HEIGHT=
"150" CLIP="10,20,40,80" Z-INDEX="1" BGCOLOR="#99FFCC">
<B>LAY1</B>
</LAYER>
```

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("clip.top：",document.layers["LAY1"].clip.
top);
document.write("<BR>");
document.write("clip.left：",document.layers["LAY1"].clip
.left);
document.write("<BR>");
document.write("clip.right：",document.layers["LAY1"].
clip.right);
document.write("<BR>");
document.write("clip.bottom：",document.layers["LAY1"].
clip.bottom);
document.write("<BR>");
document.write("clip.wide：",document.layers["LAY1"].
clip.width);
document.write("<BR>");
document.write("clip.height：",document.layers["LAY1"].
clip.height);
//-->
</SCRIPT>
</BODY>
</HTML>
```

<LAYER>→コラム「Netscape Navigator4.xのレイヤーについて・1～4」：P.204参照

上下のレイヤーの情報を取得する

JavaScript

document.layers[レイヤー名].above.name [プロパティ]
document.layers[レイヤー名].below.name [プロパティ]

*上下のレイヤーの情報を取得する

上のレイヤー名:LAY3
下のレイヤー名:LAY1

LAY1
LAY2
LAY3

解説

重なったレイヤーの1つ上のレイヤーの値は「above」プロパティで、1つ下のレイヤーの値は「below」プロパティで、参照することができます。
レイヤーは、「Z-INDEX」属性を設定していない時、HTMLでより下に記述されているレイヤーが上に重なっていきます。サンプルでは、真ん中のレイヤー「LAY2」から「above」プロパティと「below」プロパティを使用して、上のレイヤー「LAY3」と下のレイヤー「LAY 1」のオブジェクト名を、「name」プロパティで取得しています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*上下のレイヤーの情報を取得する<P>
<LAYER NAME="LAY1" LEFT="200" TOP="50" WIDTH="200" HEIGHT
="150" BGCOLOR="#99FFCC">
<B>LAY1</B>
</LAYER>
```

```
<LAYER NAME="LAY2" LEFT="225" TOP="75" WIDTH="200" HEIGHT
="150" BGCOLOR="Cyan">
<B>LAY2</B>
</LAYER>
<LAYER NAME="LAY3" LEFT="250" TOP="100" WIDTH="200" HEI
GHT="150" BGCOLOR="Brown">
<B>LAY3</B>
</LAYER>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write(" 上のレイヤー名：",document.layers["LAY2"].
above.name);
document.write("<BR>");
document.write(" 下のレイヤー名：",document.layers["LAY2"].
below.name);
//-->
</SCRIPT>
</BODY>
</HTML>
```

<LAYER>→コラム「Netscape Navigator4.xのレイヤーについて・1～4」:P.204参照

JavaScript

レイヤーの表示・非表示を切り替える

document.layers[*レイヤー名*].visibility="*show* | *hide*" [プロパティ]

Before

*レイヤーの表示・非表示を切り替える

Show!! Hide!!

After

*レイヤーの表示・非表示を切り替える

Show!! Hide!!

解説

サンプルでは、レイヤーを使ってウインドウ上にレイアウトされているボタンをクリックすることによって、同じようにレイヤーでレイアウトされている画像を、見える状態(Show)にしたり、隠して見えない状態(hide)にしたりしています。
「hide」は「hidden」と記述しても、大丈夫です。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function SImg(){ document.layers['LAY1'].visibility='show' }
function HImg(){ document.layers['LAY1'].visibility='hide' }
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*レイヤーの表示・非表示を切り替える<P>
<LAYER NAME="LAY1" VISIBILITY="hide" TOP=50 LEFT=200>
   <IMG SRC="image.gif" ALT="image.gif" WIDTH=100 HEIGHT
=100>
</LAYER>
<LAYER NAME="LAY2" VISIBILITY="show" TOP=180 LEFT=185>
  <FORM>
    <INPUT TYPE="button" NAME="showimag" VALUE="Show!!"
onClick="SImg()">
   <INPUT TYPE="button" NAME="hideimag" VALUE="Hide!!"
onClick="HImg()">
  </FORM>
</LAYER>
</BODY></HTML>
```

<LAYER>→コラム「Netscape Navigator4.xのレイヤーについて・1～4」:P.204参照

レイヤーを移動する

document.layers[*レイヤー名*].left=*ピクセル* [プロパティ]

*レイヤーを移動する

*レイヤーを移動する

*レイヤーを移動する

*レイヤーを移動する

Welcome To MY Home Page

解説

サンプルでは、レイヤーによって重ねられた文字列の内、「Welcome To」の文字列を左に、「Home Page」の文字列を右に、10ピクセルずつ200ミリ秒ごとに移動させています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function MVon(){
    if ( document.layers['LAY1'].left >10 ){ document.
layers['LAY1'].left -= 10  }
```

```
    if( document.layers['LAY3'].left < 370 ){ document.
layers['LAY3'].left += 10 }
}
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*レイヤーを移動する<P>
<LAYER NAME="LAY1" VISIBILITY="show" TOP=50 LEFT=200>
   <FONT SIZE=7><B><I>Welcome To </I></B><FONT>
</LAYER>
<LAYER NAME="LAY2" VISIBILITY="show" TOP=50 LEFT=290>
   <FONT SIZE=7><B><I>MY</I></B><FONT>
</LAYER>
<LAYER NAME="LAY3" VISIBILITY="show" TOP=50 LEFT=200>
   <FONT SIZE=7><B><I>Home Page</I></B><FONT>
</LAYER>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
setInterval("MVon()", 200);
//-->
</SCRIPT>
</BODY>
</HTML>
```

<LAYER>→コラム「Netscape Navigator4.xのレイヤーについて・1～4」:P.204参照

レイヤーの背景色を変える

NN4.0 NN4.06

bgColor 【プロパティ】

Before	After
*レイヤーの背景色を変える ここにマウスを持ってくるとバックの色が変わります...	*レイヤーの背景色を変える ここにマウスを持ってくるとバックの色が変わります...

解説

<LAYER>タグ内にJavaScriptを記述することにより、その階層のレイヤーのみに影響するJavaScriptを設定することができます。
サンプルでは、レイヤー上にマウスカーソルを乗せたり、レイヤーからマウスカーソルが離れた時、イベントが発生してレイヤーのバックの色が変わります。

Sample

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*レイヤーの背景色を変える<P>
<LAYER NAME="layer1" VISIBILITY="show" BGCOLOR="blue"
TOP=50 LEFT=100 WIDTH=200 HEIGHT=200>
    <LAYER NAME="layer2" BGCOLOR="blue" TOP=50 LEFT=50
WIDTH=100 HEIGHT=100 onMouseOver="changeColor('red')"on
MouseOut="changeColor('blue')">
        <P>ここにマウスを持ってくるとバックの色が変わります...</P>
        <SCRIPT LANGUAGE="JavaScript1.2">
        <!--
        function changeColor(NC) { bgColor=NC }
        //-->
        </SCRIPT>
    </LAYER>
</LAYER>
</BODY>
</HTML>
```

<LAYER>→コラム「Netscape Navigator4.xのレイヤーについて・1〜4」:P.204参照
巻末付録「カラーチャート1〜3」:巻末参照

スタイルシートの情報を取得する

【オブジェクトの作成】

```
document.all[スタイルシート名].style
document.all.スタイルシート名
document.スタイルシート名
document.all[添番]

document.all[スタイルシート名].style.left          [プロパティ]
document.all[スタイルシート名].style.top           [プロパティ]
document.all[スタイルシート名].pixelLeft           [プロパティ]
document.all[スタイルシート名].style.pixelTop      [プロパティ]
document.all[スタイルシート名].style.zIndex        [プロパティ]
document.all[スタイルシート名].style.visibility    [プロパティ]
```

*スタイルシートの情報を取得する

STY1... 画面左からの位置:
200px
画面上からの位置:50px
画面左からの位置(数値):200
画面上からの位置(数値):50
Z-Indexの値:2
可視属性:visible

解説

Internet Explorerでのスタイルシートはドキュメントオブジェクトのプロパティなので、オブジェクトの参照はサンプルのように「document.all[スタイルシート名].style」・「document.all.スタイルシート名」・「document.スタイルシート名」、又はオブジェクトを配列として取り扱って「document.all[添番]」とすることにより可能です。ただし、「document.all.スタイルシート名」の用法は、Windows版のInternet Explorerではエラーになります。スタイルシート名を囲む「[ ]」は、「( )」でも構いません。

「left」プロパティは、ページあるいは親スタイルシートの左上角からのスタイルシート左上角のX軸上の位置の値を、「top」プロパティは、ページあるいは親スタイルシートの左上角からのスタイルシート左上角のX軸上の位置の値を、文字列で返します。

「pixelLeft」プロパティは、ページあるいは親スタイルシートの左上角からのスタイルシート左上角のX軸上の位置の値を、「pixelTop」プロパティは、ページあるいは親スタイルシートの左上角からのスタイルシート左上角のX軸上の位置の値を、数値で返します。

また、「zIndex」プロパティはスタイルシートのzインデックスの値を、「visibility」プロパティは可視属性を返します。
これらのプロパティは変更可能なので、ページ表示後にこれらの値を変更することにより、レイヤーを移動したり、可視属性を変化することが可能です。
Internet Explorer4.x以降で使用可能です。

IE4.0
IE5.0

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*スタイルシートの情報を取得する<P>
<DIV ID="STY1" STYLE="position:absolute; left:200px; top:
50px; width:200px; height:150px; Z-INDEX=2 ;visibility:vi
sible;">
<B>STY1...</B>
</<DIV>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("画面左からの位置：",document.all["STY1"].sty
le.left) ;
document.write("<BR>") ;
document.write("画面上からの位置：",document.all.STY1.style.
top) ;
document.write("<BR>") ;
document.write("画面左からの位置(数値):",document.all.STY1.sty
le.pixelLeft) ;
document.write("<BR>") ;
document.write("画面上からの位置(数値):",document.all["STY1"].
style.pixelTop) ;
document.write("<BR>") ;
document.write("Z-Indexの値：",document.all["STY1"].style.
zIndex) ;
document.write("<BR>") ;
document.write("可視属性:",document.all["STY1"].style.visi
bility) ;
//-->
</SCRIPT>
</BODY>
</HTML>
```

<DIV >→「スタイルシート」の「任意の範囲にスタイルを適用させるためのタグ」:P.194参照

JavaScript

子スタイルシートの情報を取得する

IE4.0
IE5.0

【オブジェクトの作成】

```
document.all[スタイルシート名].all[スタイルシート名]
document.all.スタイルシート名.all.スタイルシート名

document.all[スタイルシート名].all[スタイルシート名].style.left [プロパティ]
document.all[スタイルシート名].all[スタイルシート名].style.top [プロパティ]
document.all[スタイルシート名].all[スタイルシート名].pixelLeft [プロパティ]
document.all[スタイルシート名].all[スタイルシート名].style.pixelTop
                                                         [プロパティ]
document.all[スタイルシート名].all[スタイルシート名].style.zIndex
                                                         [プロパティ]
document.all[スタイルシート名].all[スタイルシート名].style.visibility
                                                         [プロパティ]
```

*子スタイルシートの情報を取得する

親の左からの位置:50px
親の上からの位置:50px
親の左からの位置(数値):50
親の上からの位置(数値):50
Z-Indexの値:2
可視属性:visible

子スタイルシート

解説

ネストされた子スタイルシートの参照は、サンプルのように「document.all[スタイルシート名].all[スタイルシート名]」・「document.all.スタイルシート名.all.スタイルシート名」とすることによって可能です。
スタイルシート名を囲む「[ ]」は、「( )」でも構いません。
Internet Explorer4.x以降で使用可能です。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*子スタイルシートの情報を取得する<P>
<DIV ID="STY1" STYLE="position:absolute; left:200px; top
:50px; width:200px; height:150px; Z-INDEX=1 ;visibility
:visible; background: Green">
    <DIV ID="STY2" STYLE="position:absolute; left:50px;
top:50px; width:100px; height:50px; Z-INDEX=2 ;visibili
ty:visible; background: Tan">
<B>子スタイルシート</B>
    </DIV>
</DIV>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("親の左からの位置：",document.all["STY1"].all
["STY2"].style.left) ;
document.write("<BR>") ;
document.write("親の上からの位置：",document.all.STY1.all.STY
2.style.top) ;
document.write("<BR>") ;
document.write("親の左からの位置(数値):",document.all["STY1"].
all["STY2"].style.pixelLeft) ;
document.write("<BR>") ;
document.write("親の上からの位置(数値):",document.all["STY1"].
all["STY2"].style.pixelTop) ;
document.write("<BR>") ;
document.write("Z-Index の値：",document.all["STY1"].all
["STY2"].style.zIndex) ;
document.write("<BR>") ;
document.write("可視属性:",document.all["STY1"].all["STY2"].
style.visibility) ;
//-->
</SCRIPT>
</BODY>
</HTML>
```

<DIV >→「スタイルシート」の「任意の範囲にスタイルを適用させるためのタグ」:P.194参照

JavaScript

年・月・日・時・分・秒を表示する

```
オブジェクト名 = new Date()
オブジェクト名.getYear()          [メソッド]
オブジェクト名.getDate()          [メソッド]
オブジェクト名.getMonth()         [メソッド]
オブジェクト名.getHours()         [メソッド]
オブジェクト名.getMinutes()       [メソッド]
オブジェクト名.getSeconds()       [メソッド]
```

*年・月・日・時・分・秒を表示する

1999年8月24日12時55分42秒

*年・月・日・時・分・秒を表示する

1999年8月24日12時58分36秒

解説

サンプルでは、「now = new Date()」で、マシンのシステム時計から現在時刻の要素を取り出したオブジェクト「now」を作成し、メソッドを使ってそこから各時間の要素を取得しています。

「getYear()」メソッドは西暦の下2桁の数値を、「getMonth()」メソッドは1月を0とした月に対応した0から11までの数値を、「getDate()」メソッドは日に対応した1から31までの数値を、「getHours()」メソッドは時間に対応した0から23までの数値を、「getMinutes()」メソッドは分に対応した0から59までの数値を、「getSeconds()」メソッドは秒に対応した0から59までの数値を、それぞれ取得します。

Sample

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*年・月・日・時・分・秒を表示する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
now = new Date();
    document.write("19",now.getYear(),"年");
    document.write(now.getMonth()+1,"月",now.getDate(),"
日");
    document.write(now.getHours(),"時",now.getMinutes(),
"分");
    document.write(now.getSeconds(),"秒");
//--->
</SCRIPT>
</BODY></HTML>
```

JavaScript

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

午前午後を表示する

オブジェクト名 = new Date()
オブジェクト名.getHours() 【メソッド】

*午前午後を表示する

午前

解説 サンプルでは、「getHours()」メソッドで現在時刻を取得し、その数値が12より小さければ「午前」、大きければ「午後」と表示します。
「条件式 ? x:y」は、条件式が真の場合「x」の値を、偽の場合「y」の値を返します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*午前午後を表示する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
var now = new Date();
var AMPM = now.getHours();
    document.write( AMPM<12 ? "午前":"午後");
//--->
</SCRIPT>
</BODY>
</HTML>
```

注意

月の値を表示する時の注意

月の値を返す「getMonth()」メソッドは、実際の月より1小さい数値を返します。正確に月を表示させたい時には「getMonth()+1」と1を加えることを忘れないようにしてください。

JavaScript

曜日を表示する

```
オブジェクト名 = new Date()
オブジェクト名.getDay()
```

[メソッド]

*曜日を表示する
（月）

*曜日を表示する
（火）

*曜日を表示する
（水）

*曜日を表示する
（木）

*曜日を表示する
（金）

*曜日を表示する
（土）

*曜日を表示する
（日）

解説

曜日を取得する「getDay()」メソッドは、日曜日の場合は0、月曜日の場合は1、という順番で、曜日の値を0から6までの数値で取得します。
サンプルでは、取得した数値をif文で参照し、日曜日は赤い文字で、土曜日は青い文字で、その他の曜日はフォントの色指定なしで書き出しています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
 var y0 = "日";
 var y1 = "月";
 var y2 = "火";
 var y3 = "水";
 var y4 = "木";
 var y5 = "金";
 var y6 = "土";
function gety(y){
    if (y==0) { document.write( y0.fontcolor("red") ) }
    if (y==1) { document.write( y1 ) }
    if (y==2) { document.write( y2 ) }
    if (y==3) { document.write( y3 ) }
    if (y==4) { document.write( y4 ) }
    if (y==5) { document.write( y5 ) }
    if (y==6) { document.write( y6.fontcolor("blue") ) }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*曜日を表示する<P>
(
<SCRIPT LANGUAGE="JavaScript">
<!---
   day = new Date();
   gety(day.getDay());
//--->
</SCRIPT>
)
</BODY>
</HTML>
```

string.fontcolor()→「Stringオブジェクト」の「文字色を指定する」:P.477参照

JavaScript

休日を表示する

オブジェクト名 = new Date()
オブジェクト名.getMonth() [メソッド]
オブジェクト名.getDate() [メソッド]

*休日を表示する

今日は「元日」です!!明けましておめでとうございます

*休日を表示する

今日は「こどもの日」です

解説

サンプルでは、HTMLファイルがロードされた時に、関数「gethd()」内の「holiday.getMonth()+1」で月の値を、「holiday.getDate()」で日の値を取得します。それらの値を「function gethd(m,d){...}」内のif文で参照し、休日であれば休日名を書き出します。「function gethd(m,d)」のmには月の値が、dには日の値が渡されます。そして、if文内の条件式で、例えば「(m==1&&d==1)」は「月の値が1でかつ日の値が1である」ことを表わし、この条件式が真(true)の時、つまり1月1日の時に、「var」によって「hd0」と宣言された変数の値「今日は「元日」です!!明けましておめでとうございます」という文字列が書き出されます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
var hd0 = "今日は「元日」です!!明けましておめでとうございます";
var hd1 = "今日は「成人の日」です";
var hd2 = "今日は「建国記念日」です";
var hd3 = "今日は「春分の日」です";
var hd4 = "今日は「みどりの日」です";
var hd5 = "今日は「憲法記念日」です";
var hd6 = "今日は「国民の休日」です";
var hd7 = "今日は「こどもの日」です";
var hd8 = "今日は「海の日」です";
var hd9 = "今日は「敬老の日」です";
var hd10 = "今日は「秋分の日」です";
var hd11 = "今日は「体育の日」です";
var hd12 = "今日は「文化の日」です";
var hd13 = "今日は「勤労感謝の日」です";
```

```
var hd14 = "今日は「天皇誕生日」です";
function gethd(m,d){
    if (m==1&&d==1) { document.write( hd0 ) }
    if (m==1&&d==15) { document.write( hd1 ) }
    if (m==2&&d==11) { document.write( hd2 ) }
    if (m==3&&d==20) { document.write( hd3 ) }
    if (m==4&&d==29) { document.write( hd4 ) }
    if (m==5&&d==3) { document.write( hd5 ) }
    if (m==5&&d==4) { document.write( hd6 ) }
    if (m==5&&d==5) { document.write( hd7 ) }
    if (m==7&&d==20) { document.write( hd8 ) }
    if (m==9&&d==15) { document.write( hd9 ) }
    if (m==9&&d==23) { document.write( hd10 ) }
    if (m==10&&d==10) { document.write( hd11 ) }
    if (m==11&d==3) { document.write( hd12 ) }
    if (m==11&&d==23) { document.write( hd13 ) }
    if (m==12&&d==23) { document.write( hd14) }
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*休日を表示する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
   holiday = new Date();
   gethd(holiday.getMonth()+1,holiday.getDate());
//--->
</SCRIPT>
</BODY>
</HTML>
```

注意

Macintosh版のNetscapeでDateオブジェクトを使う時の注意

Macintosh版のNetscape Navigatorでは、グリニッジ標準時(GTM)を扱う「toGMTString()」や「getTimezoneOffset()」のメソッドは正しい時間を返しません。

また、「getTime()」・「parse()」・「UTC()」などの、1970年1月1日0時0分0秒からの経過時間を取り扱うメソッドも、正しい日付を返さない場合があります。

JavaScript

国際標準時やローカルタイムを表示する

```
オブジェクト名 = new Date()
オブジェクト名.toGMTString()                    [メソッド]
オブジェクト名.toLocaleString()                 [メソッド]
オブジェクト名.getTimezoneOffset()              [メソッド]
```

*国際標準時やローカルタイムを表示する

グリニッジ標準時(GTM):Mon, 2 Aug 1999 01:07:11 UTC
ローカルタイム:08/02/1999 10:07:11
グリニッジ標準時(GTM)とローカルタイムの差:-540分

解説

「toGMTString()」メソッドは日付と時間をGTM形式の文字列に変換し、「toLocaleString()」メソッドは日付と時間をローカルタイムの文字列に変換します。
「toGMTString()」は実行されたマシンの環境によって、「toLocaleString()」は実行された地域とマシンの環境よって結果が違います。
「getTimezoneOffset()」は、グリニッジ標準時とローカルタイムの差を分で返します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*国際標準時やローカルタイムを表示する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
    gtm = new Date();
    document.write("グリニッジ標準時(GTM):",gtm.toGMTString());
    document.write("<BR>");
    document.write("ローカルタイム:",gtm.toLocaleString());
    document.write("<BR>");
    document.write("グリニッジ標準時(GTM)とローカルタイムの差:",gtm.
getTimezoneOffset(),"分");
//--->
</SCRIPT>
</BODY>
</HTML>
```

日時を後から変更する

オブジェクト名 **= new Date("*month day, year hours:minutes:seconds*")**
オブジェクト名.**getTime()** 【メソッド】
オブジェクト名.**setTime()** 【メソッド】


```
*日時を後から変更する

Tue May 2 23:45:00 UTC+0900 1967
の31日後は
Fri Jun 2 23:45:00 UTC+0900 1967
```

解説

サンプルでは、「Bday = new Date("may 2, 1967 23:45:00")」で1967年5月2日23時45分の要素を持った「Bday」というオブジェクトを作り、その後「setTime()」メソッドを使用して、「Bday」オブジェクトを31日後の日時を持つオブジェクトにセットし直しています。
「setTime()」メソッドは、ミリ秒単位で日付と時間の設定を行います。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*日時を後から変更する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
Bday = new Date("may 2, 1967 23:45:00");
Day = 24*60*60*1000;
document.write(Bday);
document.write("<BR>の31日後は<BR>");
Bday.setTime(Bday.getTime() + Day * 31);
document.write(Bday);
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

年・月・日・時・分・秒を設定する

```
オブジェクト名 = new Date("month day, year hours:minutes:seconds")
オブジェクト名.setYear()                [メソッド]
オブジェクト名.setMonth()               [メソッド]
オブジェクト名.setDate()                [メソッド]
オブジェクト名.setHours()               [メソッド]
オブジェクト名.setMinutes()             [メソッド]
オブジェクト名.setSeconds()             [メソッド]
```

*年・月・日・時・分・秒を設定する

設定前:Fri May 2 23:00:00 UTC+0900 1997
年の設定変更:Sat May 2 23:00:00 UTC+0900 1970

設定前:Fri May 2 23:00:00 UTC+0900 1997
月の設定変更:Tue Sep 2 23:00:00 UTC+0900 1997

設定前:Fri May 2 23:00:00 UTC+0900 1997
日の設定変更:Tue Jun 3 23:00:00 UTC+0900 1997

解説

「setYear()」メソッドは年の下2桁を設定し、「setMonth()」メソッドは0を1月とした数値で月を設定し、「setDate()」メソッドは日を設定し、「setHours()」メソッドは時間を設定し、「setMinutes()」メソッドは分を設定し、「setSeconds()」メソッドは秒を設定します。

サンプルでは、各メソッドを使って、1度設定された年・月・日・時・分・秒の設定変更を行っています。設定変更が、その部分以外の時間の要素にも影響をあたえている(例えば、秒に60以上の数値を設定した場合、分などもそれに合わせて変更されている)点に注目してください。

Sample

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*年・月・日・時・分・秒を設定する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date("may 2, 1997 23:00:00");
document.write("設定前:",NewDay);
document.write("<BR>");
NewDay.setYear(70);
document.write("年の設定変更:",NewDay);
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
```

```
<!---
NewDay = new Date("may 2, 1997 23:00:00");
document.write("設定前:",NewDay);
document.write("<BR>");
NewDay.setMonth(8);
document.write("月の設定変更:",NewDay);
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date("may 2, 1997 23:00:00");
document.write("設定前:",NewDay);
document.write("<BR>");
NewDay.setDate(34);
document.write("日の設定変更:",NewDay);
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date("may 2, 1997 23:00:00");
document.write("設定前:",NewDay);
document.write("<BR>");
NewDay.setHours(14);
document.write("時間の設定変更:",NewDay);
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date("may 2, 1997 23:00:00");
document.write("設定前:",NewDay);
document.write("<BR>");
NewDay.setMinutes(186);
document.write("分の設定変更:",NewDay);
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date("may 2, 1997 23:00:00");
document.write("設定前:",NewDay);
document.write("<BR>");
NewDay.setSeconds(246);
document.write("秒の設定変更:",NewDay);
//--->
</SCRIPT>
</BODY></HTML>
```

NN4.0
NN4.06
IE4.0
IE5.0

JavaScript

4桁の西暦を表示する

オブジェクト名 = new Date()
オブジェクト名.getFullYear() [メソッド]

*4桁の西暦を表示する

1999年

解説

「getFullYear()」メソッドは、年の値を4桁で返します。
以前からあった「getYear()」メソッドは、基本的に西暦の下2桁の値しか返しませんが、「setFullYear()」メソッドの場合は、例えば「1999」といったように4桁すべての値を返します。
「getYear()」メソッドでも西暦2000年以降の年の値を取り扱えますが、ECMAScriptと仕様を合わせるために追加されました。
JavaScript1.3で追加されたメソッドですが、Netscape Navigator4.0や、Internet Explorer4.xでも使用することができます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*4桁の西暦を表示する<P>
<SCRIPT LANGUAGE="JavaScript">
<!--
now = new Date();
document.write(now.getFullYear(),"年");
//-->
</SCRIPT>
</BODY>
</HTML>
```

日付・時間情報を利用する

4桁の西暦を設定する

オブジェクト名 = new Date()
オブジェクト名.setFullYear()　　[メソッド]

*4桁の西暦を設定する

西暦設定前:1999年
西暦設定後:2001年

解説

「setFullYear()」メソッドは、4桁の年の値を設定します。
以前からあった「setYear()」メソッドは、基本的に年の下2桁をのみを設定しますが、「setFullYear()」メソッドは、サンプルのように年の値すべてを設定します。
「setYear()」メソッドでも西暦2000年以降の年の値を取り扱えますが、ECMAScriptと仕様を合わせるために追加されました。
JavaScript1.3で追加されたメソッドですが、Netscape Navigator4.0や、Internet Explorer4.xでも使用することができます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*4桁の西暦を設定する<P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
now = new Date();
document.write("西暦設定前:",now.getFullYear(),"年");
document.write("<BR>");
now.setFullYear(2001);
document.write("西暦設定後:",now.getFullYear(),"年");
//-->
</SCRIPT>
</BODY>
</HTML>
```

NN4.0
NN4.06
IE4.0
IE5.0

JavaScript

ミリセコンドを表示する

オブジェクト名 **= new Date()**
オブジェクト名.**getMilliseconds()** [メソッド]

*ミリセコンドを表示する

1000分の640秒

解説

「getMilliseconds()」メソッドは、1000分の1秒の値を0から999の数値で取得します。ECMAScriptと仕様を合わせるために、JavaScript1.3で追加されたメソッドですが、Netscape Navigator4.0や、Internet Explorer4.xでも使用することができます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ミリセコンドを表示する<P>
<SCRIPT LANGUAGE="JavaScript">
<!--
now = new Date();
document.write("1000分の",now.getMilliseconds(),"秒");
//-->
</SCRIPT>
</BODY>
</HTML>
```

ミリセコンドを設定する

JavaScript

オブジェクト名 = new Date()
オブジェクト名.setMilliseconds() [メソッド]

NN4.0 NN4.06 IE4.0 IE5.0

*ミリセコンドを設定する

ミリセコンド設定前:1000分の870秒
ミリセコンド設定前:1000分の500秒

解説

「setMilliseconds()」メソッドは、1000分の1秒の値を0から999の数値で設定します。ECMAScriptと仕様を合わせるために、JavaScript1.3で追加されたメソッドですが、Netscape Navigator4.0や、Internet Explorer4.xでも使用することができます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ミリセコンドを設定する<P>
<SCRIPT LANGUAGE="JavaScript">
<!--
now = new Date();
document.write(" ミリセコンド設定前：","1000 分の ",now.getMilli
seconds(),"秒");
document.write("<BR>");
now.setMilliseconds(500);
document.write(" ミリセコンド設定前：","1000 分の ",now.getMilli
seconds(),"秒");
//-->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

UTCを表示する

```
オブジェクト名 = new Date()
オブジェクト名.getUTCFullYear()                [メソッド]
オブジェクト名.getUTCMonth()                   [メソッド]
オブジェクト名.getUTCDate()                    [メソッド]
オブジェクト名.getUTCHours()                   [メソッド]
オブジェクト名.getUTCMinutes()                 [メソッド]
オブジェクト名.getUTCSeconds()                 [メソッド]
オブジェクト名.getUTCMilliseconds()            [メソッド]
オブジェクト名.getUTCDay()                     [メソッド]
```

*UTCを表示する

1999年8月2日1時12分19秒930ミリ秒(月曜日)

解説

JavaScript1.3では、UTC(Coordinated Universal Time：協定世界時)を取得する多くのメソッドが追加されています。

サンプルでは、「now = new Date()」で、マシンのシステム時計から現在時刻の要素を取り出したオブジェクト「now」を作成し、そこから各UTCの時間の要素を、メソッドを使って取得しています。

「getUTCFullYear()」メソッドは4桁の西暦の数値を、「getUTCMonth()」メソッドは1月を0とした月に対応した0から11までの数値を、「getUTCDate()」メソッドは日に対応した1から31までの数値を、「getUTCHours()」メソッドは時間に対応した0から23までの数値を、「getUTCMinutes()」メソッドは分に対応した0から59までの数値を、「getUTCSeconds()」メソッドは秒に対応した0から59までの数値を、「getUTC Milliseconds()」メソッドは1000分の1秒に対応した0から999までの数値を、「get Day()」メソッドは日曜日を0、月曜日を1、という順番で0から6までの数値を、それぞれUTCで取得します。

これらは、ECMAScriptと仕様を合わせるために、JavaScript1.3で追加されたメソッドですが、Netscape Navigator4.0や、Internet Explorer4.xでも使用することができます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!---
 var y0 = "日";
 var y1 = "月";
 var y2 = "火";
 var y3 = "水";
 var y4 = "木";
 var y5 = "金";
 var y6 = "土";
function getUTC(y){
    if (y==0) { document.write( y0.fontcolor("red") ) }
    if (y==1) { document.write( y1 ) }
    if (y==2) { document.write( y2 ) }
    if (y==3) { document.write( y3 ) }
    if (y==4) { document.write( y4 ) }
    if (y==5) { document.write( y5 ) }
    if (y==6) { document.write( y6.fontcolor("blue") ) }
}
//--->
</SCRIPT>
<BODY BGCOLOR="#FFFFFF">
*UTCを表示する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
now = new Date();
    document.write(now.getUTCFullYear(),"年");
    document.write(now.getUTCMonth()+1," 月 ",now.getUTC
Date(),"日");
    document.write(now.getUTCHours(),"時",now.getUTCMinu
tes(),"分");
    document.write(now.getUTCSeconds()," 秒 ",now.getUTC
Milliseconds(),"ミリ秒");
    document.write("(");
    getUTC(now.getUTCDay());
    document.write("曜日)");
//--->
</SCRIPT>
</BODY>
</HTML>
```

UTCを設定する

```
オブジェクト名 = new Date()
オブジェクト名.toUTCString()                  [メソッド]
オブジェクト名.setUTCFullYear()               [メソッド]
オブジェクト名.setUTCMonth()                  [メソッド]
オブジェクト名.setUTCDate()                   [メソッド]
オブジェクト名.setUTCHours()                  [メソッド]
オブジェクト名.setUTCMinutes()                [メソッド]
オブジェクト名.setUTCSeconds()                [メソッド]
オブジェクト名.setUTCMilliseconds()           [メソッド]
```

*UTCを設定する

設定前:Mon, 2 Aug 1999 01:12:46 UTC
UTCの年の設定変更:Thu, 2 Aug 2001 01:12:46 UTC

設定前:Mon, 2 Aug 1999 01:12:46 UTC
UTCの月の設定変更:Sun, 2 Jan 2000 01:12:46 UTC

設定前:Mon, 2 Aug 1999 01:12:46 UTC
UTCの日の設定変更:Tue, 10 Aug 1999 01:12:46 UTC

設定前:Mon, 2 Aug 1999 01:12:46 UTC
UTCの時間の設定変更:Mon, 2 Aug 1999 14:12:46 UTC

設定前:Mon, 2 Aug 1999 01:12:47 UTC
UTCの分の設定変更:Mon, 2 Aug 1999 04:06:47 UTC

設定前:Mon, 2 Aug 1999 01:12:47 UTC
UTCの秒の設定変更:Mon, 2 Aug 1999 01:15:00 UTC

設定前:Mon, 2 Aug 1999 01:12:47 UTC
UTCのミリセコンドの設定変更:Mon, 2 Aug 1999 01:12:49 UTC

解説

JavaScript1.3では、UTC(Coordinated Universal Time：協定世界時)を設定する多くのメソッドが追加されています。
「setUTCFullYear()」メソッドは4桁のUTCの年の値を設定し、「setUTCMonth()」メソッドは0を1月とした数値でUTCの月の値を設定し、「setUTCDate()」メソッドはUTCの日の値を設定し、「setUTCHours()」メソッドはUTCの時間の値を設定し、「setUTCMinutes()」メソッドはUTCの分の値を設定し、「setUTCSeconds()」メソッドはUTCの秒の値を設定します。

サンプルでは、ページが読み込まれた時の時間の値を、各メソッドを使って変更しています。なお、「toUTCString()」メソッドは、UTCの日付と時間を文字列へ変換します。
これらは、ECMAScriptと仕様を合わせるために、JavaScript1.3で追加されたメソッドですが、Netscape Navigator4.0や、Internet Explorer4.xでも使用することができます。

NN4.0
NN4.06
IE4.0
IE5.0

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*UTCを設定する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date();
document.write("設定前:",NewDay.toUTCString());
document.write("<BR>");
NewDay.setUTCFullYear(2001);
document.write("UTCの年の設定変更:",NewDay.toUTCString());
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date();
document.write("設定前:",NewDay.toUTCString());
document.write("<BR>");
NewDay.setUTCMonth(12);
document.write("UTCの月の設定変更:",NewDay.toUTCString());
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date();
document.write("設定前:",NewDay.toUTCString());
document.write("<BR>");
NewDay.setUTCDate(10);
document.write("UTCの日の設定変更:",NewDay.toUTCString());
//--->
</SCRIPT>
```

```
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date();
document.write("設定前:",NewDay.toUTCString());
document.write("<BR>");
NewDay.setUTCHours(14);
document.write("UTCの時間の設定変更:",NewDay.toUTCString());
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date();
document.write("設定前:",NewDay.toUTCString());
document.write("<BR>");
NewDay.setUTCMinutes(186);
document.write("UTCの分の設定変更:",NewDay.toUTCString());
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date();
document.write("設定前:",NewDay.toUTCString());
document.write("<BR>");
NewDay.setUTCSeconds(180);
document.write("UTCの秒の設定変更:",NewDay.toUTCString());
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date();
document.write("設定前:",NewDay.toUTCString());
document.write("<BR>");
NewDay.setUTCMilliseconds(2000);
document.write("UTCのミリセコンドの設定変更:",NewDay.toUTCString
());
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

日付をカウントダウンする

オブジェクト名 = new Date()
オブジェクト名 = new Date(*year*, *month*, *day*)
オブジェクト名.getTime()

[メソッド]

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

JavaScript

*日付をカウントダウンする

西暦2001年まで後518日

解説

サンプルではまず、「today = new Date()」で現在の時間の要素を持った「today」というオブジェクトと、「NextC = new Date(2001,0,1)」で西暦2001年1月1日の要素を持った「NextC」というオブジェクトを作っています。そして、「getTime()」メソッドを使って、それぞれのオブジェクトの1970年1月1日0時0分0秒からのミリ秒単位の経過時間を取得し、その差の値を「/(24*60*60*1000)」とすることによって、日数に戻しています。
「new Date(2001,0,1)」内の日付を変更することによって、希望の日付までのカウントダウンをすることができます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*日付をカウントダウンする<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
   today = new Date();
   NextC = new Date(2001,0,1);
   NC = (NextC.getTime()-today.getTime())/(24*60*60*1000)
   document.write("西暦2001年まで後"+ Math.ceil(NC) +"日")
//--->
</SCRIPT>
</BODY>
</HTML>
```

Math.ceil()→「Mathオブジェクト」の「最も近くて大きい整数を返す」:P.466参照

JavaScript

時間ごとに違ったメッセージを表示する

オブジェクト名 = new Date()
オブジェクト名.getHours()　　[メソッド]

*時間ごとに違ったメッセージを表示する

10時のメッセージ

*時間ごとに違ったメッセージを表示する

11時のメッセージ

解説

サンプルでは、HTMLファイルがロードされた時、関数「geth()」内の「h.getHours()」が時間の値を取得して、その値を「function geth(t){...}」内のif文で参照し、時間に合わせたメッセージを表示しています。
メッセージ部分に「<IMG SRC='URL' alt='xxx' WIDTH="xxx" HIGHT="xxx">」と画像を表示するタグを書けば、時間によって違った画像を表示することもできます。
サンプルのように1時間ごとに1つのメッセージを設定している場合は、本当はif文をネスト(入れ子)にする必要はありません。
数時間ごとに処理を行う方法は、「時間によって背景画像を変える」(次項)を参考にしてください。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
 var m0 = "0時のメッセージ";
 var m1 = "1時のメッセージ";
 var m2 = "2時のメッセージ";
 var m3 = "3時のメッセージ";
 var m4 = "4時のメッセージ";
 var m5 = "5時のメッセージ";
 var m6 = "6時のメッセージ";
 var m7 = "7時のメッセージ";
 var m8 = "8時のメッセージ";
 var m9 = "9時のメッセージ";
 var m10 = "10時のメッセージ";
 var m11 = "11時のメッセージ";
 var m12 = "12時のメッセージ";
 var m13 = "13時のメッセージ";
 var m14 = "14時のメッセージ";
 var m15 = "15時のメッセージ";
```

```
 var m16 = "16時のメッセージ";
 var m17 = "17時のメッセージ";
 var m18 = "18時のメッセージ";
 var m19 = "19時のメッセージ";
 var m20 = "20時のメッセージ";
 var m21 = "21時のメッセージ";
 var m22 = "22時のメッセージ";
 var m23 = "23時のメッセージ";
function geth(t){
    if (t==0) document.write( m0 );
        else { if (t==1) document.write( m1 );
        else { if (t==2) document.write( m2 );
        else { if (t==3) document.write( m3 );
        else { if (t==4) document.write( m4 );
        else { if (t==5) document.write( m5 );
        else { if (t==6) document.write( m6 );
        else { if (t==7) document.write( m7 );
        else { if (t==8) document.write( m8 );
        else { if (t==9) document.write( m9 );
        else { if (t==10) document.write( m10 );
        else { if (t==11) document.write( m11 );
        else { if (t==12) document.write( m12 );
        else { if (t==13) document.write( m13 );
        else { if (t==14) document.write( m14 );
        else { if (t==15) document.write( m15 );
        else { if (t==16) document.write( m16 );
        else { if (t==17) document.write( m17 );
        else { if (t==18) document.write( m18 );
        else { if (t==19) document.write( m19 );
        else { if (t==20) document.write( m20 );
        else { if (t==21) document.write( m21 );
        else { if (t==22) document.write( m22 );
        else { if (t==23) document.write( m23 );
    }}}}}}}}}}}}}}}}}}}}}}}
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*時間ごとに違ったメッセージを表示する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
   h = new Date();
   geth(h.getHours());
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

時間によって背景画像を変える

```
オブジェクト名 = new Date()
オブジェクト名.getHours()
```

[メソッド]

*時間によって背景画像を変える

0時~5時:BACK1.gifの画像
6時~11時:BACK2.gifの画像
12時~17時:BACK3.gifの画像
18時~23時:BACK4.gifの画像

*時間によって背景画像を変える

0時~5時:BACK1.gifの画像
6時~11時:BACK2.gifの画像
12時~17時:BACK3.gifの画像
18時~23時:BACK4.gifの画像

解説

JavaScriptには、背景の色を指定する「bgColor」プロパティはありますが、背景画像を指定するコマンドが、今の所ありません。
サンプルでは、その対策として、</HEAD>と<BODY>の間に、時間に応じた背景画像の指定を含む<BODY>タグを書き出すようにしています。
<BODY>タグの中に画像だけでなく、フォントの色などを指定しても評価されます。
しかし、JavaScript対応のブラウザでは、<BODY>タグが2つあることになるので、HTMLの文法的にはあまり正しいとはいえません。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function geth(t){
    if (t<=5) document.write("<BODY BACKGROUND='BACK1.
gif'>");
        else { if (t<=11) document.write("<BODY BACKGROU
ND='BACK2.gif'>");
        else { if (t<=17) document.write("<BODY BACKGROU
ND='BACK3.gif'>");
        else { if (t<=23) document.write("<BODY BACKGROU
ND='BACK4.gif'>");
    }}}
}
//--->
```

```
</SCRIPT>
</HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!---
h = new Date();
   geth(h.getHours());
//--->
</SCRIPT>
<BODY BGCOLOR="#FFFFFF">
*時間によって背景画像を変える<P>
0時~5時:BACK1.gifの画像<BR>
6時~11時:BACK2.gifの画像<BR>
12時~17時:BACK3.gifの画像<BR>
18時~23時:BACK4.gifの画像
</BODY>
</HTML>
```

<BODY>→「ページの基礎となる内容」の「HTMLに最低限必要な要素」:P.29参照

<BODY BACKGROUND='URL'>→「ページの基礎となる内容」の「背景画像を指定する」:P.31参照

リアルタイムに年・月・日を表示する

JavaScript

オブジェクト名 = new Date()
オブジェクト名.getYear() [メソッド]
オブジェクト名.getMonth() [メソッド]
オブジェクト名.getDate() [メソッド]

*リアルタイムに年・月・日を表示する

1999.08.17

*リアルタイムに年・月・日を表示する

1999.08.24

解説

サンプルでは、Dateオブジェクトの「getYear()」メソッドで年を、「getMonth()」メソッドで月を、「getDate()」メソッドで日を取得し、フォーム内に書き出す処理を0.5秒ごとに繰り返しています。
フォームに書き出す時、年は西暦の下2桁を取得するのでその前に「19」を付ける処理を、月は取得した数値に1を加えて10以下の時には頭に0を付ける処理を、日は取得した数値が10以下の時には頭に0を付ける処理をしています。
また、Netscape Navigator2.0のメモリーエラー対策として、10分後にスクリプトを停止する処理をしています。もし不要な場合は、「//10分後にタイマーを止める処理」のコメントがある部分の1行を削除してください。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
var TC = 0;
function DayWatch() {
    if (TC < 1200) {     //10分後にタイマーを止める処理
    TC++ ;               //10分後にタイマーを止める処理
    var day = new Date();
    var year = day.getYear();
    var month = day.getMonth()+1;
    var date = day.getDate();
        if (month < 10) {     //月.日が一桁の時頭に0を付ける処理
            month = "0" + month;
```

```
                    }
        if (date < 10) {
            date = "0" + date;
                    }
    document.Watch1.watch1.value = "19"+year +"."+month
+"."+date;
    setTimeout("DayWatch()", 500);
        }                //10分後にタイマーを止める処理
}
//---->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF" onLoad="DayWatch()">
*リアルタイムに年・月・日を表示する<P>
<FORM NAME="Watch1" >
<INPUT TYPE="text" NAME="watch1" SIZE=10>
</FORM>
</BODY>
</HTML>
```

---

setTimeout()→「windowオブジェクト」の「ステータス行に文字を流す」:P.292参照

clearTimeout()→「windowオブジェクト」の「ステータス行に文字を流す」:P.292参照

<INPUT TYPE="text">→「Formオブジェクト」の「フォームに文字を流す」:P.376参照

JavaScript

リアルタイムに時・分・秒を表示する

オブジェクト名 = new Date()
オブジェクト名.getHours() [メソッド]
オブジェクト名.getMinutes() [メソッド]
オブジェクト名.getSeconds() [メソッド]

*リアルタイムに時・分・秒を表示する

12:56:39

*リアルタイムに時・分・秒を表示する

12:59:00

解説

サンプルでは、Dateオブジェクトの「getHours()」メソッドで時間を、「getMinutes()」メソッドで分を、「getSeconds()」メソッドで秒を取得し、フォーム内に書き出す処理を0.5秒ごとに繰り返しています。
フォームに値を書き出す時、取得した各数値が10以下の場合、頭に0を付ける処理をしています。
また、Netscape Navigator2.0のメモリーエラー対策として、10分後にスクリプトを停止する処理をしています。もし不要な場合は、「//10分後にタイマーを止める処理」のコメントがある部分の1行を削除してください。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
var TC = 0 ;
function TimeWatch() {
   if (TC < 1200) {     //10分後にタイマーを止める処理
        TC++ ;          //10分後にタイマーを止める処理
    var time = new Date();
    var hour = time.getHours();
    var min = time.getMinutes();
    var sec = time.getSeconds();
```

```
            if (hour < 10) { //時.分.秒が一桁の時頭に0を付ける処理
                hour = "0" + hour;
                            }
            if (min < 10) {
                min = "0" + min;
                           }
            if (sec < 10) {
                sec = "0" + sec;
                           }
        document.Watch2.watch2.value = hour+':'+min+':'+sec;
        setTimeout("TimeWatch()", 500);
                     }      //10分後にタイマーを止める処理
    }
    //--->
    </SCRIPT>
    </HEAD>
    <BODY BGCOLOR="#FFFFFF" onLoad="TimeWatch()">
    *リアルタイムに時・分・秒を表示する<P>
    <FORM NAME="Watch2" >
    <INPUT TYPE="text" NAME="watch2" SIZE=10>
    </FORM>
    </BODY>
    </HTML>
```

---

setTimeout()→「windowオブジェクト」の「ステータス行に文字を流す」:P.292参照

clearTimeout()→「windowオブジェクト」の「ステータス行に文字を流す」:P.292参照

<INPUT TYPE="text">→「Formオブジェクト」の「フォームに文字を流す」:P.376参照

JavaScript

自然対数の底

Math.E [プロパティ]

*自然対数の底

2.718281828459045

解説

「E」プロパティは、自然対数の底(オイラー定数)を返します。
このプロパティは、読み出し専用です。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.E)
//--->
</SCRIPT>
```

JavaScript

eを底とする2の自然対数

Math.LN2 [プロパティ]

*eを底とする2の自然対数

0.6931471805599453

解説

「LN2」プロパティは、eを底とする2の自然対数の底を返します。
このプロパティは、読み出し専用です。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.LN2)
//--->
</SCRIPT>
```

JavaScript

eを底とする10の自然対数

Math.LN10 **[プロパティ]**

*eを底とする10の自然対数

2.302585092994046

解説 「LN10」プロパティは、eを底とする10の自然対数の底を返します。
このプロパティは、読み出し専用です。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.LN10)
//--->
</SCRIPT>
```

JavaScript

2を底とする自然対数

Math.LOG2E **[プロパティ]**

*2を底とする自然対数

1.4426950408889633

解説 「LOG2E」プロパティは、2を底とする自然対数を返します。
このプロパティは、読み出し専用です。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.LOG2E)
//--->
</SCRIPT>
```

JavaScript

10を底とする自然対数

Math.LOG10E [プロパティ]

*10を底とする自然対数

0.4342944819032518

解説

「LOG10E」プロパティは、10を底とする自然対数を返します。
このプロパティは、読み出し専用です。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.LOG10E)
//--->
</SCRIPT>
```

JavaScript

円周率

Math.PI [プロパティ]

*円周率

3.141592653589793

解説

「PI」プロパティは、円周率を返します。
このプロパティは、読み出し専用です。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.PI)
//--->
</SCRIPT>
```

JavaScript

1/2の平方根

Math.SQRT1_2 [プロパティ]

*1/2の平方根

0.7071067811865476

解説

「SQRT1_2」プロパティは、ルート2分の1の値を返します。
このプロパティは、読み出し専用です。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.SQRT1_2)
//--->
</SCRIPT>
```

JavaScript

2の平方根

Math.SQRT2 [プロパティ]

*2の平方根

1.4142135623730951

解説

「SQRT2」プロパティは、ルート2の値を返します。
このプロパティは、読み出し専用です。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.SQRT2)
//--->
</SCRIPT>
```

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

JavaScript

n,mを比較して小さい方の数値を返す

Math.min(*n*,*m*) [メソッド]

*n,mを比較して小さい方の数値を返す

1

解説 「min(n,m)」メソッドは、nとmを比較して小さい方の数値を返します。「max()」メソッドの逆の働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.min(1,5))
//--->
</SCRIPT>
```

JavaScript

n,mを比較して大きい方の数値を返す

Math.max(*n*,*m*) [メソッド]

*n,mを比較して大きい方の数値を返す

5

解説 「max(n,m)」メソッドは、nとmを比較して大きい方の数値を返します。「min()」メソッドの逆の働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.max(1,5))
//--->
</SCRIPT>
```

JavaScript

最も近くて小さい整数を返す

Math.floor(*n*) 【メソッド】

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

*最も近くて小さい整数を返す

5
-6

解説

「floor()」メソッドは、nに最も近くて小さい整数を返します。
「ceil()」メソッド(次項)の逆の働きをします。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*最も近くて小さい整数を返す<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.floor(5.2));
document.write("<BR>");
document.write(Math.floor(-5.2));
//--->
</SCRIPT>
</BODY>
</HTML>
```

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript

最も近くて大きい整数を返す

Math.ceil(*n*) [メソッド]

```
*最も近くて大きい整数を返す

6
-5
```

解説

「ceil(n)」メソッドは、nにもっとも近くて大きい整数を返します。
「floor()」メソッド(前項)の逆の働きをします。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*最も近くて大きい整数を返す<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.ceil(5.2));
document.write("<BR>");
document.write(Math.ceil(-5.2));
//--->
</SCRIPT>
</BODY>
</HTML>
```

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript

0からの絶対値を返す

Math.abs(*n*) [メソッド]

*0からの絶対値を返す

5
5

解説

「abs(n)」メソッドは、nの0からの絶対値(0からの距離)を返します。
サンプルの通り、「-5」も「5」も「5」となります。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.abs(-5));
document.write("<BR>");
document.write(Math.abs(5));
//--->
</SCRIPT>
```

JavaScript

四捨五入した数値を返す

Math.round(*n*) [メソッド]

*四捨五入した数値を返す

5
-6

解説

「round(n)」メソッドは、nの小数第一位を四捨五入した数値を返します。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.round(5.49));
document.write("<BR>");
document.write(Math.round(-5.64));
//--->
</SCRIPT>
```

JavaScript

乱数を発生させておみくじを作る

Math.random() [メソッド]

*乱数を発生させておみくじを作る

あなたの運勢は: 凶

*乱数を発生させておみくじを作る

あなたの運勢は: 末吉

*乱数を発生させておみくじを作る

あなたの運勢は: 吉

*乱数を発生させておみくじを作る

あなたの運勢は: 大吉

解説

「0から1までの乱数を返します。
サンプルでは、「Math.random()」で発生した乱数の値を、関数「Unsei()」内の処理で参照し、その値によって書き出す文字を変更することによって、ページがロードされるごとにランダムに表示される文字が変わるおみくじを作っています。
Netscape Navigator2.xではUNIX版のみ作動します。3.0以降のすべてのOSのNetscape Navigator及び、Windows版のInternet Explorer3.0以降のブラウザでは正常に作動します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
 var Omikuzi0 = "大吉";
 var Omikuzi1 = "中吉";
 var Omikuzi2 = "小吉";
 var Omikuzi3 = "吉";
 var Omikuzi4 = "末吉";
 var Omikuzi5 = "凶";
 var Omikuzi6 = "大凶";
```

```
function Unsei(n){
    if (n<=0.15) document.write( Omikuzi0.bold() );
        else { if (n<=0.3) document.write( Omikuzi1.bold() );
        else { if (n<=0.45) document.write( Omikuzi2.bold() );
        else { if (n<=0.6) document.write( Omikuzi3.bold() );
        else { if (n<=0.75) document.write( Omikuzi4.bold() );
        else { if (n<=0.9) document.write( Omikuzi5.bold() );
        else { if (n<=1) document.write( Omikuzi6.bold() );
    }}}}}}
}
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*乱数を発生させておみくじを作る<P>
あなたの運勢は：
<SCRIPT LANGUAGE="JavaScript">
<!---
Unsei(Math.random())
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

nのm乗を返す

Math.pow(*n*,*m*) [メソッド]

*nのm乗を返す

25

解説

「pow(n,m)」メソッドは、nのm乗の数値を返します。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.pow(5,2))
//--->
</SCRIPT>
```

JavaScript

平方根を返す

Math.sqrt(*n*) [メソッド]

*平方根を返す

1.4142135623730951

解説

「sqrt(n)」メソッドは、nの平方根の値を返します。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.sqrt(2))
//--->
```

JavaScript

対数を返す

Math.exp(*n*) [メソッド]

*対数を返す

148.4131591025766

解説

「exp(n)」メソッドは、e(オイラー定数)をn乗した数値を返します。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.exp(5))
//--->
</SCRIPT>
```

JavaScript

自然対数を返す

Math.log(*n*) [メソッド]

*自然対数を返す

1.6094379124341002

解説

「log(n)」メソッドは、nの自然対数を返します。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.log(5))
//--->
</SCRIPT>
```

JavaScript

サインを返す

Math.sin(*n*) [メソッド]

*サイン を返す

0.479425538604203

解説

「sin(n)」メソッドは、nのサイン(正弦)の値を返します。
単位はラジアンです。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.sin(0.5))
//--->
</SCRIPT>
```

JavaScript

コサインを返す

Math.cos(*n*) [メソッド]

*コサインを返す

0.8775825618903728

解説

「cos(n)」メソッドは、nのコサイン(余弦)の値を返します。
単位はラジアンです。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.cos(0.5))
//--->
</SCRIPT>
```

JavaScript

タンジェントを返す

Math.tan(*n*) [メソッド]

*タンジェントを返す

0.5463024898437905

解説

「tan(n)」メソッドは、nのタンジェント(正接)の値を返します。
単位はラジアンです。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.tan(0.5))
//--->
</SCRIPT>
```

JavaScript

x,y座標から角度を返す

Math.atan2(*x*,*y*) [メソッド]

*x,y座標から角度を返す

45度

解説

「atan2(x,y)」メソッドは、x,y座標から角度を求めます。
単位はラジアンです。
サンプルでは、ラジアンから度を求めています。
ドキュメント化はJavaScript1.1からですが、JavaScript1.0でも使用できます。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.atan2(5,5) * 180 / Math.PI,"度")
//--->
</SCRIPT>
```

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript

アークサインを返す

Math.asin(*n*) [メソッド]

*アークサインを返す

0.5235987755982989

解説

「asin(n)」メソッドは、nのアークサイン(逆正弦)の値を返します。
nは-1から1までの数値が入り、それ以外の場合には0を返します。
単位はラジアンです。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.asin(0.5))
//--->
</SCRIPT>
```

JavaScript

アークコサインを返す

Math.acos(*n*) [メソッド]

*アークコサインを返す

1.0471975511965976

解説

「acos(n)」メソッドは、nのアークコサイン(逆余弦)の値を返します。
nは-1から1までの数値が入り、それ以外の場合には0を返します。
単位はラジアンです。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.acos(0.5))
//--->
</SCRIPT>
```

アークタンジェントを返す

Math.atan(*n*) **[メソッド]**

*アークタンジェントを返す

0.4636476090008061

解説

「atan(n)」メソッドは、nのアークタンジェント(逆正接)の値を返します。単位はラジアンです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*アークタンジェントを返す<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write(Math.atan(0.5))
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

文字を大きくする

文字列.**big()** [メソッド]

*文字を大きくする

文字を大きくする

解説

「big()」メソッドは、文字列を通常より大きな文字で表示します。
<BIG>タグ(P.64参照)と同じ働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("文字を大きくする".big())
//--->
</SCRIPT>
```

JavaScript

文字を小さくする

文字列.**small()** [メソッド]

*文字を小さくする

文字を小さくする

解説

「small()」メソッドは、文字列を通常より小さな文字で表示します。
<SMALL>タグ(P.64参照)と同じ働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("文字を小さくする".small())
//--->
</SCRIPT>
```

文字色を指定する

文字列.fontcolor("色指定") **[メソッド]**

*文字色を指定する

文字色を指定する
文字色を指定する

解説

「fontcolor」メソッドは、文字列の色を指定します。
<FONT COLOR="色指定">タグ(P.67参照)と同じ働きをします。
色指定は、色の名前か16進数で設定します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*文字色を指定する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("文字色を指定する".fontcolor("green"));
document.write("<BR>");
document.write("文字色を指定する".fontcolor("FF0000"));
//--->
</SCRIPT>
</BODY>
</HTML>
```

<FONT COLOR="色指定">→「スタイルとレイアウト」の「文字色を指定する」:P.67参照
巻末付録「カラーチャート1～3」:巻末参照

JavaScript

フォントのサイズを指定する

文字列.fontsize(*n*) [メソッド]

*フォントのサイズを指定する

フォントサイズを指定する

解説

「fontsize()」メソッドは、文字列のフォントサイズを指定します。
<FONT SIZE=n>タグ(P.62参照)と同じ働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("フォントサイズを指定する".fontsize(5))
//--->
</SCRIPT>
```

JavaScript

文字を点滅する

文字列.blink() [メソッド]

*文字を点滅する

*文字を点滅する

文字を点滅する

解説

「blink()」メソッドは、文字列を点滅させます。
<BLINK>タグと同じ働きをします。
Internet Explorerは、<BLINK>タグに未対応なため何も反応しません。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("文字を点滅する".blink())
//--->
</SCRIPT>
```

太字(ボールド)にする

文字列.bold()　　[メソッド]

*文字を太字(ボールド)にする

太字にする

解説

「bold()」メソッドは、文字列を太文字(ボールド)にします。
<B>タグ(P.60参照)と同じ働きをします。

Sample

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*文字を太字(ボールド)にする<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("太字にする".bold())
//--->
</SCRIPT>
</BODY>
</HTML>
```

<B>→「スタイルとレイアウト」の「フォントスタイルを指定する」:P60参照

Tip

文字列に複数の効果をあたえたい時は

文字列を修飾する時、複数の効果を同時に出したい場合は、文字列の後にその効果をあたえるメソッドを並べて記述します。

例えば文字列を、太文字にして色を付け、斜体文字にする場合は、スクリプトタグ内で次のように記述します。

```
document.write("文字列".bold().fontcolor("green").italics())
```

ちなみに、このスクリプトは、HTMLで次のように記述するのと同じです。

```
<B><FONT COLOR="green"><I>文字列</I></FONT></B>
```

JavaScript

削除文字にする

文字列.strike() [メソッド]

*削除文字にする

~~削除文字にする~~

解説

「strike()」メソッドは、文字列を削除文字にします。
<STRIKE>タグ(P.60参照)と同じ働きをします。

Sample

```
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*削除文字にする<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("削除文字にする".strike())
//--->
</SCRIPT>
</BODY>
</HTML>
```

<STRIKE>→「スタイルとレイアウト」の「フォントスタイルを指定する」:P.60参照

JavaScript

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

上付き文字にする

文字列.sup()　[メソッド]

*上付き文字にする

上付き文字にする

Sample

「sup()」メソッドは、文字列を上付文字にします。
<SUP>タグ(P.60参照)と同じ働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("上付き文字にする".sup())
//--->
</SCRIPT>
```

JavaScript

下付き文字にする

文字列.sub()　[メソッド]

*下付き文字にする

下付き文字にする

Sample

「sub()」メソッドは、文字列を下付文字にします。
<SUB>タグ(P.60参照)と同じ働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("下付き文字にする".sub())
//--->
</SCRIPT>
```

JavaScript

斜体文字(イタリック)にする

文字列.italics()　　[メソッド]

*斜体文字(イタリック)にする

斜体文字にする

解説

「italics()」メソッドは、文字列を斜体文字にします。
<I>タグ(P.60参照)と同じ働きをします。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*斜体文字(イタリック)にする<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("斜体文字にする".italics())
//--->
</SCRIPT>
</BODY>
</HTML>
```

<I>→「スタイルとレイアウト」の「フォントスタイルを指定する」:P.60参照

重要

日本語(2バイト文字)の取り扱いについて

Netscape Navigatorでは、JavaScript1.2までは文字を常に1バイト文字として認識しているので、日本語のひらがなや漢字などの2バイト文字は1文字を2字と認識していました。それに対してInternet Explorer(日本語版)は、1バイト文字も2バイト文字も1文字は1字として認識します。

このため、日本語に対して「indexOf()」メソッドや「substring()」メソッドなど、文字を検索したり抜き出すスクリプトを使用したり、「length」プロパティで文字の数を調べたりすると、Netscape NavigatorとInternet Explorer(日本語版)では異なった結果になりました。

JavaScript1.3からは、文字コードがUNICODEになったので、Netscape Navigatorでも日本語も1文字を1字としてカウントするようになりました。

JavaScript

等幅文字にする

文字列.fixed()　　[メソッド]

*等幅文字にする

等幅文字にする

解説

「fixed()」メソッドは、文字列を等幅フォントで表示します。
<TT>タグ(P.60参照)と同じ働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("等幅文字にする".fixed())
//--->
</SCRIPT>
```

JavaScript

リンクを作る

文字列.link("URL")　　[メソッド]

*リンクを作る

リンクを作成する

解説

「link()」メソッドは、リンクを設定します。
<A HREF ="URL">タグ(P.76参照)と同じ働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("リンクを作成する".link("http://www.netscape.
com/"))
//--->
</SCRIPT>
```

JavaScript

アンカーを設定する

文字列.anchor() [メソッド]

*アンカーを設定する

アンカーの設定

解説

「anchor()」メソッドは、アンカーの設定をします。
<A NAME="文字列">タグ(P.78参照)と同じ働きをします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("アンカーの設定".anchor("test1"))
//--->
</SCRIPT>
```

JavaScript

大文字を小文字に変換する

文字列.toLowerCase() [メソッド]

*大文字を小文字に変換する

oomoziwokomozinisuru

解説

「toLowerCase()」メソッドは、大文字の文字列を小文字に変換します。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("OOMOZIWOKOMOZINISURU".toLowerCase())
//--->
</SCRIPT>
```

JavaScript

小文字を大文字に変換する

文字列.toUpperCase()　　[メソッド]

*小文字を大文字に変換する

KOMOZIWOOOMOZINISURU

解説

「toUpperCase()」メソッドは、小文字の文字列を大文字に変換します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*小文字を大文字に変換する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
document.write("komoziwooomozinisuru".toUpperCase())
//--->
</SCRIPT>
</BODY>
</HTML>
```

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

NN3.0
NN4.0
NN4.06

IE4.0
IE5.0

JavaScript

文字列を分割する

文字列.split("分割する文字") [メソッド]

*文字列を分割する

Welcome / To / My / Home / Page /
My

解説

「split()」メソッドは、文字列を指定した文字の部分で分割します。
サンプルでは、「Welcome To My Home Page」の各スペースごとに文字列を分割し、「/」を挿入しています。
また、分割された文字列は[0]から始まる配列になります。したがって、「SPmozi[2]」は、「My」を返します。
JavaScript1.1で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*文字列を分割する<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
var mozi="Welcome To My Home Page";
var space=" ";
var SPmozi = mozi.split(space);
    for (var i=0; i < SPmozi.length; i++) {
        document.write (SPmozi[i] + " / ");
    }
document.write ("<BR>");
document.write (SPmozi[2]);
//--->
</SCRIPT>
</BODY>
</HTML>
```

文字列を操作する

n番目の文字を抜き出す

文字列.**charAt(*n*)** ［メソッド］

*n番目の文字を抜き出す

T
ム

解説

「charAt(n)」メソッドは、0スタートの先頭の文字から数えてn番目の文字を抜き出します。
サンプルでは、「Welcome To My Home Page」の9番目の文字「T」が抜き出されます。
Netscape Navigatorの場合、JavaScript1.2までは、日本語のような2バイト文字は1文字を2字として見ていたため、日本語の文字列を正確に抜き出すことができませんでした。しかし、JavaScript1.3からは、文字コードがUNICODEになったので、日本語も正確に抜き出されます。
なお、Internet Explorerは、日本語も半角英数字も1文字としてカウントします。

Sample

```
<SCRIPT LANGUAGE="JavaScript">
<!---
var mozi1 = "Welcome To My Home Page"
var mozi2 = "ようこそ私のホームページへ"
    document.write(mozi1.charAt(8))
    document.write("<BR>")
    document.write(mozi2.charAt(8))
//--->
</SCRIPT>
```

Tip

文字列以外も検索・文字修飾する方法

stringオブジェクトのメソッドは、""で括られた文字列以外でも、オブジェクトも検索したり、文字修飾の効果をあたえることができます。

例えば<SCRIPT>タグ内で、「navigator.appVersion.charAt(0)」と記述した場合は、使用しているブラウザのバージョンの1番始めの数字を返し、「document.write(document.lastModified.italics())」と記述した場合は、ファイルの最終更新日時が斜体文字で書き出されます。

これは、各オブジェクトがstringの属性を持っているためです。

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript

文字列の途中の文字を抜き出す

文字列.substring(*n*,*m*) **[メソッド]**

*文字列の途中の文字を抜き出す

To M
ムページ

解説

「substring()」メソッドは、先頭から数えて、n番目の文字からm番目の文字の直前までの文字列を抜き出します。
Netscape Navigatorの場合は、JavaScript1.2までは、日本語のような2バイト文字は1文字を2字としてカウントしていましたが、JavaScript1.3からは文字コードがUNICODEになったので、日本語も1文字を1字としてカウントするようになっています。Internet Explorerでは、日本語も半角英数字も1文字としてカウントします。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*文字列の途中の文字を抜き出す<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
var mozi1 = "Welcome To My Home Page";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.substring(8,12));
    document.write("<BR>");
    document.write(mozi2.substring(8,12));
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

n番からm個の文字を抜き出す

NN4.0
NN4.06
IE4.0
IE5.0

文字列.**substr(*n*,*m*)** [メソッド]

*n番からm個の文字を抜き出す

To M
ムページ

解説

「substr()」メソッドは、0スタートの先頭から数えて、n番目の文字からm個の文字を抜き出します。n番目からm番目の文字を抜き出す、JavaScript1.0の「substring()」(「文字列の途中の文字を抜き出す」：左ページ)との違いに注意してください。
Netscape Navigatorの場合は、JavaScript1.2までは、日本語のような2バイト文字は1文字を2字としてカウントしていましたが、JavaScript1.3からは文字コードがUNICODEになったので、日本語も1文字を1字としてカウントするようになっています。Internet Explorerでは、日本語も半角英数字も1文字としてカウントします。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*n番からm個の文字を抜き出す<P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
var mozi1 = "Welcome To My Home Page"
var mozi2 = "ようこそ私のホームページへ"
    document.write(mozi1.substr(8,4));
    document.write("<BR>");
    document.write(mozi2.substr(8,4));
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

先頭から文字列を検索する

文字列.indexOf("検索対象文字",[検索開始位置]) [メソッド]

```
*先頭から文字列を検索する

5
4
-1
```

解説

「indexOf()」メソッドは、文字列を先頭から検索します。
「"検索対象文字"」で指定した文字の位置を、0スタートの先頭から数えた数字で返し、もし指定した文字が含まれていない場合は、「-1」を返します。
Netscape Navigatorの場合は、JavaScript1.2までは、日本語のような2バイト文字は1文字を2字としてカウントしていました。JavaScript1.3からは、文字コードがUNICODEになったので、日本語も1文字を1字としてカウントするようになっています。Internet Explorerでは、日本語も半角英数字も1文字としてカウントします。
また、「[検索開始位置]」は省略可能で、その場合は先頭から検索します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*先頭から文字列を検索する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
var mozi1 = "Welcome To My Home Page";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.indexOf("m"));
    document.write("<BR>");
    document.write(mozi2.indexOf("私",0));
    document.write("<BR>");
    document.write(mozi1.indexOf("A",0));
//-->
</SCRIPT>
</BODY>
</HTML>
```

後ろから文字列を検索する

文字列.lastIndexOf("*検索対象文字*",[*検索開始位置*])　　[メソッド]


```
*後ろから文字列を検索する

16
4
-1
```

解説

「lastIndexOf()」メソッドは、文字列を後ろから検索します。
「検索対象文字」で指定した文字の位置を、0スタートの先頭から数えた数字で返し、もし指定した文字が含まれていない場合は、「-1」を返します。
Netscape Navigatorの場合は、JavaScript1.2までは、日本語のような2バイト文字は1文字を2字としてカウントしていました。JavaScript1.3からは、文字コードがUNICODEになったので、日本語も1文字を1字としてカウントするようになっています。Internet Explorerでは、日本語も半角英数字も1文字としてカウントします。
また、「[検索開始位置]」は省略可能で、その場合は1番後ろから検索します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*後ろから文字列を検索する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
var mozi1 = "Welcome To My Home TPage";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.lastIndexOf("m"));
    document.write("<BR>");
    document.write(mozi2.lastIndexOf("私",12));
    document.write("<BR>");
    document.write(mozi1.lastIndexOf("A"));
//--->
</SCRIPT>
</BODY>
</HTML>
```

NN4.0
NN4.06
IE4.0
IE5.0

JavaScript

指定した文字のISO-Latin-1のコード番号を返す

String.charCodeAt(*n*) [メソッド]

*指定した文字のISO-Latin-1のコード番号を返す

66

解説

「charCodeAt()」メソッドは、スタートの先頭から数えて、n番目の文字をISO-Latin-1の文字コードに変換します。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*指定した文字のISO-Latin-1のコード番号を返す<P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("ABC".charCodeAt(1))
//-->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

ISO-Latin-1のコード番号を文字に変換する

String.fromCharCode(要素1,要素2,...,要素n)　　[メソッド]

NN4.0
NN4.06
IE4.0
IE5.0

*ISO-Latin-1のコード番号を文字に変換する

ABC

解説

「fromCharCode()」メソッドは、ISO-Latin-1のコードを文字に変換します。サンプルのように複数のコードを「,」で区切って、1度に指定できます。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ISO-Latin-1のコード番号を文字に変換する<P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write(String.fromCharCode(65,66,67))
//-->
</SCRIPT>
</BODY>
</HTML>
```

Tip

ISO-Latin-1コードとASCIIコード

ISO-Latin-1コードは「0」から「255」まであり、そのうち「0」から「127」はASCIIのコードセットと共通しています。

したがって「KeyDown」や「KeyPress」、「KeyUp」などのイベントが発生した時に、イベントが発生したキーのASCIIの値を取得できるeventオブジェクトのプロパティ「which」を使用する場合、「String.fromCharCode(KeyDown.which)」といった感じで「fromCharCode()」と併せて使用すると、ASCIIコードを英数字に変換することができます。

ASCIIコードは、コンピュータ用の英数字コード体系として広く普及しているコード体系で、7bit、128文字が割り当てられています。

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

曜日を表示する - Array オブジェクトを使う -

オブジェクト名 = **new Array(***配列の数***)**

*曜日を表示する - Arrayオブジェクトを使う -
（月）

*曜日を表示する - Arrayオブジェクトを使う -
（火）

*曜日を表示する - Arrayオブジェクトを使う -
（水）

*曜日を表示する - Arrayオブジェクトを使う -
（木）

*曜日を表示する - Arrayオブジェクトを使う -
（金）

*曜日を表示する - Arrayオブジェクトを使う -
（土）

*曜日を表示する - Arrayオブジェクトを使う -
（日）

解説

サンプルでは、「Dateオブジェクト」(P.432～)で説明した、曜日を表示するスクリプト「曜日を表示する」(P.434)を、Arrayオブジェクトを使って書き直したものです。「youbi= new Array(7)」で日曜から土曜までの7つの配列要素を持った「youbi」というオブジェクトを作成して、「getDay()」で取得した値と照合し、値が「0」の時は「youbi[0]」である「日」を、値が「1」の時は「youbi[1]」である「月」を…といった具合に配列の要素から文字列を取り出しています。
arrayオブジェクトはJavaScript1.1で追加されたオブジェクトですが、このサンプルは、Netscape Navigator2.xでも使用できます。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
youbi= new Array(7)
youbi[0] = "日";
youbi[1] = "月";
youbi[2] = "火";
youbi[3] = "水";
youbi[4] = "木";
youbi[5] = "金";
youbi[6] = "土";
function gety(y){ document.write( youbi[y] ) }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*曜日を表示する(Arrayオブジェクトを使う)<P>
(
<SCRIPT LANGUAGE="JavaScript">
<!---
day = new Date();
gety(day.getDay());
//--->
</SCRIPT>
)
</BODY>
</HTML>
```

new Date()→「Dateオブジェクト」の「年・月・日・時・分・秒を表示する」:P.432参照

getDay()→「Dateオブジェクト」の「曜日を表示する」:P.434参照

Tip

Netscape2.0でも使えるArrayオブジェクトは?

Areaオブジェクトは、Netscape Navigator3.0(JavaScript1.1)から追加されたオブジェクトですが、「arrayオブジェクト名 = new Array(配列の数)」の用法でのみ、Netscape Navigator2.0でも使用可能です。

しかし、それ例外の「arrayオブジェクト名 = new Array(0番目の要素,...,n番目の要素)」で作成された配列のオブジェクトや、「join()」・「reverse()」・「sort()」などのメソッドは使用できません。

JavaScript

配列の要素を文字列にして書き出す

オブジェクト名 = new Array(0番目の要素,1番目の要素, ...,n番目の要素)
オブジェクト名.join() [メソッド]

*配列の要素を文字列にして書き出す

0番目/1番目/2番目/3番目

解説

「join()」メソッドは、配列の要素を文字列に変換します。
「join("文字列")」と指定すると、要素の区切り時にカッコ内の文字列が加えられます。
何も指定がない時には、各要素は「,」で区切られます。
JavaScript1.1で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*配列の要素を文字列にして書き出す<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
hairetu= new Array("0番目","1番目","2番目","3番目");
document.write(hairetu.join("/"));
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

配列の要素を逆に並べ変える

オブジェクト名 = new Array(0番目の要素,1番目の要素, ...,n番目の要素)
オブジェクト名.reverse() [メソッド]

NN3.0 NN4.0 NN4.06 IE4.0 IE5.0

*配列の要素を逆に並べ変える

3番目,2番目,1番目,0番目

解説

「reverse()」メソッドは、配列の要素を逆に並べ変えます。
JavaScript1.1で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*配列の要素を逆に並べ変える<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
hairetu= new Array("0番目","1番目","2番目","3番目");
document.write(hairetu.reverse());
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

配列の要素をソートする

オブジェクト名 = new Array(0番目の要素,1番目の要素, ...,n番目の要素)
オブジェクト名.sort(比較関数) 【メソッド】

*配列の要素をソートする

辞書順:array,form,location,window

昇順:8,32,62,256

降順:256,62,32,8

解説

「sort()」メソッドは、配列の要素をソートします。
比較関数があたえられていない時は、各配列の要素が文字列として扱われ、辞書編集法にしたがってソートされます。
また、数値を比較する時は、比較関数を「function 関数名(a, b){ return a - b }」とすると昇順ソートに、「function 関数名(a, b){ return b - a }」とすると降順ソートになります。
JavaScript1.1で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*配列の要素をソートする<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
Nhairetu1= new Array("location","array","window","form");
document.write("辞書順:",Nhairetu1.sort());
//--->
</SCRIPT>
<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
Nhairetu2= new Array(62,8,256,32);
function SortSet(a,b) { return a - b }
document.write("昇順:",Nhairetu2.sort(SortSet));
//--->
</SCRIPT>
```

```
<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
Nhairetu2= new Array(62,8,256,32);
function SortSet(a,b) { return b - a }
document.write("降順:",Nhairetu2.sort(SortSet));
//--->
</SCRIPT>
</BODY>
</HTML>
```


注意

Sort()を使用する時の注意

「Sort()」を使って配列をソートした結果を、連続して「document.write」で書き出そうとすると一部、あるいは全部が表示されない場合があります。

これは、配列を評価しソートするのに時間がかかり、結果を返すより前にページのロードが終ってしまうためだと思われます。

マシンパワーにも左右されるようなので、「Sort()」を使う時には、色々な環境でチェックすることをお勧めします。

この問題は、Netscape Navigator4.xでは解消されています。

JavaScript

新しい関数を作る

オブジェクト名= new Function ([要素1,要素2,...要素n], functionの働き)
オブジェクト名.length 【プロパティ】
オブジェクト名.arity 【プロパティ】

```
*新しい関数を作る

16*32 = 512
length:2
arity:2
```

解説

サンプルでは、new演算子を使って「x」と「y」2つの要素を乗算する関数「myFunc」を作り、そこに数値を入れて計算しています。
また、「length」プロパティ・「arity」プロパティは、要素数を値に持っています。「length」はJavaScript1.1、「arity」はJavaScript1.2で追加されたプロパティです。
現在は、Internet Explorerでは「arity」をサポートしていません。

Sample

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*新しい関数を作る<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
myFunc = new Function("x", "y", "return x * y");
document.write("16*32 = " + myFunc(16,32));
document.write("<BR>");
document.write("length:" + myFunc.length);
//--->
</SCRIPT>
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
document.write("<BR>");
document.write("arity:" + myFunc.arity);
//--->
</SCRIPT>
</BODY>
</HTML>
```

関数がどこから呼ばれたか参照する

オブジェクト名.caller　【プロパティ】

NN3.0 NN4.0 NN4.06 IE4.0

Before

*関数がどこから呼ばれたか参照する

クリックしてください

この文字の上にマウスカーソルを持ってきてください

After

[JavaScript アプリケーション]

このファンクションは
function onclick(event) {
myFunc();
}
から呼ばれました

OK

*関数がどこから呼ばれたか参照する

クリックしてください

この文字の上にマウスカーソルを持ってきてください

[JavaScript アプリケーション]

このファンクションは
function onmouseover(event) {
myFunc();
}
から呼ばれました

OK

解説

「caller」プロパティは、どの関数がどこから呼ばれたかを取得します。
サンプルでは、フォームのボタンをクリックしたり、リンクの上にマウスカーソルが乗った時などに発生した関数名と共に、関数を発生させたイベントのタイプを表示した警告用のダイアログボックスを開きます。
同様の働きをするプロパティに、オブジェクトの作成元、つまり関数名を取得する「constructor」プロパティがあります。
JavaScript1.1で追加されたプロパティです。
このスクリプトは、Internet Explorer5.0ではうまく機能しない場合があります。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.1">
<!----
function myFunc() {
    if (myFunc.caller == null) {
        alert("このファンクションはフォームのTopから呼ばれました");
    }
   else { alert(" このファンクションは " + myFunc.caller + " から
呼ばれました");
   }
}
//--->
```

```
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*関数がどこから呼ばれたか参照する<P>
<FORM>
<INPUT TYPE="button" NAME= "FCall" VALUE="クリックしてください"
 onClick="myFunc()" >
</FORM>
<P>
<A HREF="PT.html" onMouseOver="myFunc()"> この文字の上にマウス
カーソルを持ってきてください</A>
</BODY>
</HTML>
```

alert()→「windowオブジェクト」の「警告用のダイアログボックスを開く」:P.287参照

関数の内容を配列として使用する

オブジェクト(ファンクション)名.**arguments.length** 【プロパティ】
オブジェクト(ファンクション)名.**arguments.callee** 【プロパティ】
オブジェクト(ファンクション)名.**arguments** 【インデックス】

*関数の内容を配列として使用する

- One
- Two
- Three

関数の内容:function list() { for (var i=0; i" + list.arguments[i]); } document.write("
"); document.write("関数の内容:" + list.arguments.callee); }

解説

「arguments」は、それ自体が関数の配列を作成するオブジェクトで、JavaScript1.0から使用可能でしたが、JavaScript1.1からはFunctionオブジェクトのプロパティに加えられました。
「arguments」は、「length」プロパティを持っています。
サンプルでは、「list.arguments.length」で関数「list()」の要素数を取得し、for文を使って「list.arguments[0]」の要素から順番に書き出しています。また、「callee」プロパティは、関数の内容を値に持っています。
「length」はJavaScript1.1、「callee」はJavaScript1.2で追加されたプロパティです

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
function list() {
    for (var i=0; i<list.arguments.length; i++) {
        document.write("<LI>" + list.arguments[i]);
    }
document.write("<BR>");
document.write("関数の内容:" + list.arguments.callee);
}
//--->
</SCRIPT>
</HEAD>
```

NN3.0
NN4.0
NN4.06
IE4.0
IE5.0


```
<BODY BGCOLOR="#FFFFFF">
*関数の内容を配列として使用する<P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
list("One","Two","Three")
//--->
</SCRIPT>
</BODY>
</HTML>
```

コラム

JavaScript対応のソフト

JavaScriptが標準化されてから、JavaScriptに対応したアプリケーションが、ぼちぼちと発表され始めてきました。

ブラウザでは、どちらも軽量であることを売りとした「Opera」や「iCab」などがJavaScriptに対応していたり、今後対応する予定です。また、アドビシステムズ社の「Adobe Acrobat」などは、フォームのチェックをJavaScriptを使って行うことができます。

面白いものでは、ゲーム専用機のDreamcastが用意しているブラウザ「Dream passport2」で、これは、JavaScript1.1レベルのJavaScriptをサポートしています。

これからも、オープンソースで開発が進められているレンダリングエンジン「Gecko」を採用するなどした、意外なJavaScript対応のソフトが発表されていくことでしょう。

・Opera Software
http://www.opera.com/index.html

・Alexander Clauss & iCab Company
http://www.icab.de/index.html

・アドビ システムズ 株式会社：Adobe Acrobat 4.0
http://www.adobe.co.jp/product/acrobat/index.html

・株式会社 セガ・エンタープライゼス：Dreamcast
http://www.sega.co.jp/dreamcast/dreamcast.html

異なるオブジェクトを呼び出す

NN3.0
NN4.0
NN4.06

call(this, 引数,引数,...) [メソッド]

Before

*異なるオブジェクトを呼び出す

After

[JavaScript アプリケーション]

6

OK

解説

「call ()」メソッドは、引数を指定して、異なるオブジェクトを呼び出します。
サンプルでは、「x」と「y」の2つの要素を乗算する関数であるオブジェクト「myFunc」を、関数「myFunc2()」から「call ()」メソッドを使って呼び出しています。
この他にも、引数の配列と共にオブジェクトを呼び出す「apply()」メソッドがあり、両方ともJavaScript1.3で追加されたメソッドです。
現在このメソッドは、Internet Explorer5.xではサポートされていないようです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.3">
<!---
myFunc = new Function("x", "y", "return x * y");
function myFunc2(a,b) { window.alert( myFunc.call(this,
a, b) ) }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*異なるオブジェクトを呼び出す<P>
<SCRIPT LANGUAGE="JavaScript1.3">
<!---
myFunc2(2,3)
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

新しいオブジェクトを作る - 1 -

オブジェクト名 = new Object()

*新しいオブジェクトを作る - 1 -

新しいオブジェクトです

解説

JavaScriptでは、新たにユーザー独自のオブジェクトを作成することができます。サンプルでは、new演算子を使って、「新しいオブジェクトです」という文字列の値を持った新たなオブジェクト、「myObj」を作成しています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*新しいオブジェクトを作る - 1 - <P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
myObj = new Object("新しいオブジェクトです");
document.write(myObj.bold());
//--->
</SCRIPT>
</BODY>
</HTML>
```

新しいオブジェクトを作る - 2 -

オブジェクト名 = {プロパティ1:値1, プロパティs2:値2,..., プロパティn:値n}

*新しいオブジェクトを作る - 2 -

新しいオブジェクトです

解説

JavaScript1.2から、あらかじめ用意されているオブジェクトを使用するか、new演算子を使用して新しいオブジェクトを作成する以外に、サンプルの用法でも独自のオブジェクトを作成できるようになりました。
サンプルでは、文字の色やサイズの値をプロパティに持った、「mystring」というオブジェクトを作成しています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*新しいオブジェクトを作る - 2 - <P>
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
myObj = new Object("新しいオブジェクトです");
mystring = { color:"red",size:5 };
document.write(myObj.fontcolor(mystring.color).fontsize
(mystring.size));
//--->
</SCRIPT>
</BODY>
</HTML>
```

プロパティを監視する

watch(プロパティ,関数) [メソッド]
unwatch(プロパティ) [メソッド]

＊プロパティを監視する

[JavaScript アプリケーション]
myobj.proを1 から2へ変更しました
OK

[JavaScript アプリケーション]
myobj.proを2 から変わったよ!!へ変更しました
OK

解説

「watch()」メソッドは、プロパティの監視を行うメソッドです。
サンプルでは、新たに作成した「myobj」オブジェクトの「pro」プロパティを、「watch()」メソッドで監視しています。「pro」プロパティには、始め「1」の値が設定されていますが、「watch()」メソッド内で呼び出す関数の処理で「1」から「2」へ、「2」から「変わったよ!!」という文字列へと変更されていきます。ちなみに、この時の関数の引数は、「function (プロパティ,変更前の値,変更後の値)」となります。
「unwatch()」メソッドは、「watch()」メソッドで設定したプロパティの監視を解除するメソッドです。このため、「myobj.unwatch('pro')」の後に設定した「myobj.pro = 100」の処理は、行われません。
JavaScript1.3で追加されたメソッドですが、現在、Internet Explorer5.xではサポートされていないようです。

Sample

```
<SCRIPT LANGUAGE="JavaScript1.3">
<!---
myobj = {pro:1}
myobj.watch("pro",
   function (id,oldval,newval) {
      window.alert("myobj." + id + "を" + oldval + " から
" + newval + "へ変更しました")
      return newval
   })
myobj.pro = 2
myobj.pro = "変わったよ!!"
myobj.unwatch('pro')
myobj.pro = 100
//--->
</SCRIPT>
```

真(true)・偽(false)の値を設定する

オブジェクト名 = new Boolean()


```
*真(true)・偽(false)の値を設定する

true
false
true
false
```

解説

Booleanオブジェクトを使うと、明示的に「true」と「false」の値をもったオブジェクトを作成することができます。
サンプルでは、Booleanオブジェクトを使って「true」の値を持った「myTrue」オブジェクトと、「false」の値を持った「myFalse」オブジェクトを作成しています。
JavaScript1.1で追加されたオブジェクトです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*真(true)・偽(false)の値を設定する<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
myTrue = new Boolean(true);
myFalse = new Boolean(false);
document.write( myTrue == true );
document.write("<BR>");
document.write( myTrue == false );
document.write("<BR>");
document.write( myFalse != true );
document.write("<BR>");
document.write( myFalse != false );
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

NN3.0
NN4.0
NN4.06
IE4.0
IE5.0

数値を作成する

オブジェクト名 = new Number()
オブジェクト名.NaN [プロパティ]

```
*数値を作成する

100
false
true
```

解説

Numberオブジェクトを使うと、数値の値を持ったオブジェクトを作成することができます。
サンプルでは、Numberオブジェクトを使って、数値の値を持ったオブジェクト「myNumber」を作成しています。また、「NaN」プロパティを使うと、オブジェクトを数値でない状態に変更することができます。
JavaScript1.1で追加されたプロパティですが、JavaScript3.0では「NaN」が機能しない場合があります。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*数値を作成する<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
myNumber = new Number(100);
document.write(myNumber);
document.write("<BR>");
document.write(isNaN(myNumber));
document.write("<BR>");
document.write(isNaN(myNumber.NaN));
//--->
</SCRIPT>
</BODY>
</HTML>
```

isNaN()→「ビルトイン関数(top level関数」の「数値かどうかを調べる」:P.531参照

JavaScript

使用可能な数値の範囲を調べる

オブジェクト名.**MAX_VALUE** 【プロパティ】
オブジェクト名.**MIN_VALUE** 【プロパティ】
オブジェクト名.**NEGATIVE_INFINITY** 【プロパティ】
オブジェクト名.**POSITIVE_INFINITY** 【プロパティ】

NN3.0 NN4.0 NN4.06 IE4.0 IE5.0

*使用可能な数値の範囲を調べる

最大値:1.7976931348623157e+308
最小値:5e-324
負の無限大:-Infinity
無限大:Infinity

解説

Numberオブジェクトには、数値の有効範囲を取り扱う多くのプロパティがあります。「MAX_VALUE」プロパティはJavaScriptで扱える値の最大値を、「MIN_VALUE」プロパティはJavaScriptで扱える値の最小値を、「NEGATIVE_INFINITY」プロパティは負の無限大の値を、「POSITIVE_INFINITY」は無限大の値をそれぞれ持っています。これらは、JavaScript1.1で追加されたプロパティです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*使用可能な数値の範囲を調べる<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
document.write("最大値:" + Number.MAX_VALUE);
document.write("<BR>");
document.write("最小値:" + Number.MIN_VALUE);
document.write("<BR>");
document.write("負の無限大:" + Number.NEGATIVE_INFINITY);
document.write("<BR>");
document.write("無限大:" + Number.POSITIVE_INFINITY);
//--->
</SCRIPT>
</BODY>
</HTML>
```

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

JavaScript

オブジェクト(配列)の数を取得する

length [プロパティ]

【サポートしているオブジェクト(配列)】
Button・Checkbox・FileUpload・Form・frame・Hidden・Image・layer・Password・Plugin・Radio・Reset・Select・Submit・Text・Textarea・windowオブジェクト
anchors・applets・elements(フォーム内の値)・embeds・forms・frames・history・images・layers・links・mineTypes・options・plugins配列

*オブジェクト(配列)の数を取得する

1ページ目 2ページ目 3ページ目

文字の数:13
ヒストリーの数:83
リンクの数:3
フォームの数:1

解説

「length」プロパティは、HTMLファイル内のオブジェクトや配列の数を取得します。

Sample

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*オブジェクト(配列)の数を取得する<P>
<FORM NAME="kasu1">
<INPUT TYPE="text" NAME="X" SIZE=10>
</FORM>
<BR>
<A HREF="page1.html">1ページ目</A>
:<A HREF="page2.html">2ページ目</A>
:<A HREF="page3.html">3ページ目</A>
<HR>
<SCRIPT LANGUAGE="JavaScript">
<!----
 var mozi = "この文字は何文字でしょうか"
    document.write("文字の数:",mozi.length,"<BR>")
    document.write("ヒストリーの数:",history.length,"<BR>")
    document.write("リンクの数:",document.links.length,"<BR>")
    document.write(" フォームの数:",document.forms.length,"<BR>")
//--->
</SCRIPT>
</BODY></HTML>
```

オブジェクトに名前を付ける

name 【プロパティ】

【サポートしているオブジェクト】
Button・Checkbox・FileUpload・Form・frame・Hidden・Image・layer・Password・Plugin・Radio・Reset・Select・Submit・Text・Textarea・windowオブジェクト

*オブジェクトに名前を付ける

フォームエレメントの値

イメージのURL(NAME):file://Pro_server/souko/4-複数使用ToU/01_02/image.gif
イメージのURL(配列):file://Pro_server/souko/4-複数使用ToU/01_02/image.gif
フォームの内容(NAME):フォームエレメントの値
フォームの内容(配列):フォームエレメントの値

解説

Link関連のオブジェクト以外のHTMLタグを使用するオブジェクトでは、タグ内に「name」プロパティを設定することによってオブジェクトに名前を付けて、明示的にオブジェクトを参照することができるようになります。
サンプルでは、Imageオブジェクトとformオブジェクトに「name」プロパティで名前を設定することによって、配列と同じように、オブジェクトの色々な値を参照しています。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*オブジェクトに名前を付ける<P>
<IMG SRC="image.gif" NAME="kao" ALT="kao" WIDTH="110" HE
IGHT="110">
<BR>
<FORM NAME="NAIYOU">
<INPUT TYPE="text" NAME="naiyou" SIZE=30 VALUE="フォームエレ
メントの値">
</FORM>
```

```
<P>
<SCRIPT Language="JavaScript1.1">
<!---
document.write("イメージのURL(NAME):");
document.write(document.kao.src);
document.write("<BR>");
document.write("イメージのURL(配列):");
document.write(document.images[0].src);
document.write("<BR>");
document.write("フォームの内容(NAME):");
document.write(document.NAIYOU.naiyou.value);
document.write("<BR>");
document.write("フォームの内容(配列):");
document.write(document.forms[0].elements[0].value);
//---->
</SCRIPT>
</BODY>
</HTML>
```

新たにプロパティを作成する

prototype 【プロパティ】
constructor 【プロパティ】

【サポートしているオブジェクト】
Array・Boolean・Date・function・Image・Number・Select・option・stringオブジェクト、ユーザー作成オブジェクト

*新たにプロパティを作成する

ただ今の日時は: 08/02/1999 11:17:59
オブジェクトの作成元: function Date() { [native code] }

解説

「prototype」プロパティは、new演算子で作られたオブジェクトに、明示的にプロパティを追加します。
サンプルでは、new演算子で作られたtodayオブジェクトに対して、「prototype」プロパティを使って「newpro」というプロパテイを新たに作成しています。
また、「constructor」プロパティは、関数名も含めたオブジェクトの作成元の値を持っています。
あくまでも作成元の情報だけなので、そのオブジェクトがどのような値を持っているかを知るには、「toString()」メソッド(「オブジェクトを文字列に変える」:次項)や「toSource()」メソッド(「オブジェクト内の値を文字列にする」:P.519)を使う必要があります。
JavaScript1.1で追加されたプロパティです。

Sample

```
<HTML>
<HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*新たにプロパティを作成する<P>
<SCRIPT Language="JavaScript1.1">
<!---
today = new Date();
Date.prototype.newpro="ただ今の日時は:".bold();
document.write(today.newpro + today.toLocaleString());
document.write("<BR>");
document.write("オブジェクトの作成元:" + today.constructor);
//---->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

オブジェクトを文字列に変える

toString() **[メソッド]**

【サポートしているオブジェクト】
すべてのオブジェクト

```
*オブジェクトを文字列に変える

月,火,水,木,金,土
Mon Aug 2 11:21:31 UTC+0900 1999
file://Pro_server/souko/urata/4-複数使用ToU/01_05/05pro.html
文字列にすべき内容がない時:[object]
```

解説

「toString()」メソッドは、オブジェクトを文字列に変換します。
Netscape Navigator2.0でこのメソッドを使用すると、一部のブラウザでは不安定になる場合があります。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*オブジェクトを文字列に変える<P>
<SCRIPT Language="JavaScript">
<!---
var YOUBI = new Array("月","火","水","木","金","土")
document.write(YOUBI.toString() + "<BR>")
var today = new Date()
document.write(today.toString() + "<BR>")
document.write(location.toString() + "<BR>")
document.write("文字列にすべき内容がない時:"+window.toString() +
"<BR>")
//--->
</SCRIPT>
</BODY>
</HTML>
```

n進数に変換する

toString(*n*) [メソッド]


```
*n進数に変換する

10進数: 0 2進数: 0 16進数: 0
10進数: 1 2進数: 1 16進数: 1
10進数: 2 2進数: 10 16進数: 2
10進数: 3 2進数: 11 16進数: 3
10進数: 4 2進数: 100 16進数: 4
10進数: 5 2進数: 101 16進数: 5
10進数: 6 2進数: 110 16進数: 6
10進数: 7 2進数: 111 16進数: 7
10進数: 8 2進数: 1000 16進数: 8
10進数: 9 2進数: 1001 16進数: 9
10進数: 10 2進数: 1010 16進数: a
10進数: 11 2進数: 1011 16進数: b
10進数: 12 2進数: 1100 16進数: c
10進数: 13 2進数: 1101 16進数: d
10進数: 14 2進数: 1110 16進数: e
10進数: 15 2進数: 1111 16進数: f
10進数: 16 2進数: 10000 16進数: 10
```

解説 「toString()」メソッドで「toString(n)」のように数値をあたえると、n進数の数値を返します。

Sample

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*n進数に変換する<P>
<SCRIPT LANGUAGE="JavaScript">
<!----
for (x = 0; x < 17; x++) {
   document.write("10進数: ", x.toString(10), " 2進数: ",
x.toString(2), " 16進数: ", x.toString(16),"<BR>");
}
//---->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

オブジェクト内の値を返す

valueOf() **[メソッド]**

【サポートしているオブジェクト】
すべてのオブジェクト

*オブジェクト内の値を返す

月,火,水,木,金,土
935382606910
file:///C|/WINDOWS/????????/0823/4/01_07/07pro.html
返すべき数値がない時:[object Window]

解説

「valueOf()」メソッドは、オブジェクト内の値を取得します。
JavaScript1.1で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*オブジェクト内の値を返す<P>
<SCRIPT LANGUAGE="JavaScript1.1">
<!---
var YOUBI = new Array("月","火","水","木","金","土")
document.write(YOUBI.valueOf() + "<BR>")
var today = new Date()
document.write(today.valueOf() + "<BR>")
document.write(location.valueOf() + "<BR>")
document.write("返すべき数値がない時:"+window.valueOf() + "<BR>")
//--->
</SCRIPT>
</BODY>
</HTML>
```

オブジェクト内の値を文字列にする

toSource() **【メソッド】**

【サポートしているオブジェクト】
すべてのオブジェクト

```
*オブジェクト内の値を文字列にする

配列「HAIRETU」内の値:["1","2","3","4","5","6"]

Dateオブジェクト内の値:(new Date(933561778490))

locationオブジェクト内の値:
{href:"file:////PRO_SERVER/souko/urata/4-¥x95¥xA1¥x90¥x94¥x8Eg¥x97pToU/01_14/14pro.html",
protocol:"file:", host:"", hostname:"", port:"",
pathname:"//PRO_SERVER/souko/urata/4-¥x95¥xA1¥x90¥x94¥x8Eg¥x97pToU/01_14/14pro.html",
hash:"", search:"", target:null, text:null, x:undefined, y:undefined}

windowオブジェクト内の値:#1={length:0, frames:#1#, parent:#1#, top:#1#, self:#1#, name:"",
status:undefined, defaultStatus:"", opener:null, closed:false, innerWidth:723, innerHeight:452,
outerWidth:735, outerHeight:580, screenX:260, screenY:23, pageXOffset:0, pageYOffset:0, secure:false,
frameRate:60, offscreenBuffering:"auto",
document:{location:#2={href:"file:////PRO_SERVER/souko/urata/4-¥x95¥xA1¥x90¥x94¥x8Eg¥x97pToU/01_
protocol:"file:", host:"", hostname:"", port:"",
pathname:"//PRO_SERVER/souko/urata/4-¥x95¥xA1¥x90¥x94¥x8Eg¥x97pToU/01_14/14pro.html",
hash:"", search:"", target:null, text:null, x:undefined, y:undefined}}, history:{length:2, current:"",
previous:"", next:""}, location:#2#, crypto:{}, pkcs11:{}, HAIRETU:["1","2","3","4","5","6"],
today:(new Date(933561778490))}
```

解説

「toSource()」メソッドは、オブジェクト内の値を文字列に変換します。
「toString()」メソッドでは、「[object Window]」のような形式で表されていた値の詳細も取得することができます。
JavaScript1.3で追加されたメソッドです。

Sample

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*オブジェクト内の値を文字列にする<P>
<SCRIPT Language="JavaScript1.3">
<!---
var HAIRETU = new Array("1","2","3","4","5","6")
document.write("配列「HAIRETU」内の値:"+HAIRETU.toSource() + 
"<P>")
var today = new Date()
document.write("Date オブジェクト内の値:"+today.toSource() + 
"<P>")
document.write("locationオブジェクト内の値:"+location.toSource
() + "<P>")
document.write("windowオブジェクト内の値:"+window.toSource())
//--->
</SCRIPT>
</BODY></HTML>
```

JavaScript

フォームに入力された文字列を計算できるようにする

eval() 【メソッド】

【サポートしているオブジェクト】
すべてのオブジェクト

Before

*フォームに入力された文字列を計算できるようにする

2 + 6 =

計算!! Clear

After

*フォームに入力された文字列を計算できるようにする

2 + 6 = 8

計算!! Clear

解説

「eval()」メソッドは、文字列を数値に変換します。
フォーム内の値は文字列ですので、フォームに入力されたデータを計算する時は、「eval()」メソッドを使って数値に変更してから計算する必要があります。
JavaScript1.0ではビルトイン関数としてオブジェクトと結び付けられていましたが、JavaScript1.1からはメソッドになりました。

Sample

```
<HTML>
<HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
function calc(CL) { CL.Z.value = eval(CL.X.value)+eval
(CL.Y.value) }
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*フォームに入力された文字列を計算できるようにする<P>
<FORM NAME="Calc">
<INPUT TYPE="text" NAME="X" VALUE="0" SIZE=10>
+
<INPUT TYPE="text" NAME="Y" VALUE="0" SIZE=10>
=
<INPUT TYPE="text" NAME="Z" SIZE=10>
<P>
<INPUT TYPE="button" VALUE="計算!!" onClick="calc(this.form)">
<INPUT TYPE="reset" VALUE="Clear">
</FORM>
</BODY>
</HTML>
```

ウィンドウ(フレーム)をプリントする

オブジェクト名.print() **[メソッド]**

【サポートしているオブジェクト】
window･frameオブジェクト

Before

*ウィンドウ(フレーム)をプリントする

Print!!

After

印刷

プリンタ
プリンタ名(N): OKI MICROLINE 803PSII+F　プロパティ(P)...
状態: 通常使うプリンタ: 要調査 - 印刷待ちドキュメント: 0 個
種類: OKI MICROLINE 803PSII+F
場所: ¥¥Dellserver¥okimicro2　ファイルへ出力(L)
コメント:

印刷範囲
すべて(A)
ページ指定(G) 1 ページから(F)
ページまで(T)
選択した部分(S)

印刷部数
部数(C): 1
部単位で印刷(O)

OK　キャンセル

解説

「print()」メソッドは、指定されているページをプリントします。
サンプルでは、ブラウザのプリントボタンを押すのと同等の働きをするボタンを作成しています。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ウィンドウ(フレーム)をプリントする<P>
<FORM>
    <INPUT TYPE="button" NAME="print" VALUE="Print!!" 
onClick="window.print()">
</FORM>
</BODY>
</HTML>
```

<INPUT TYPE="button">→「Formオブジェクト」の「ボタンをリンクに使う」:P.372参照

一定時間ごとに処理を繰り返す

setInterval(処理,時間設定) 【メソッド】
clearInterval() 【メソッド】

【サポートしているオブジェクト】
window・frameオブジェクト

解説

「setInterval()」メソッドは、指定した時間後に1度だけ処理を行う「setTimeout()」メソッドと違い、指定された処理をミリ秒単位で繰り返し行います。
サンプルは、「Imageオブジェクト」の「アニメーションにスタート・ストップボタンを付ける」(P.404)を、「setInterval()」を使って書き直したものです。「setInterval()」を停止する時は、「clearInterval()」メソッドを使って、「setTimeout()」の用法と同じように「clearInterval(ID名)」とIDを設定して使用します。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT Language="JavaScript1.1">
<!--
var ImageSetA = 1 ;
ANIMA = new Array() ;
for(i = 1; i < 6; i++) {
   ANIMA[i] = new Image() ;
   ANIMA[i].src = "image" + i + ".gif" ;
}
function anime1() {
   document.animation.src = ANIMA[ImageSetA].src ;
   ImageSetA++ ;
       if( ImageSetA > 5) {
       ImageSetA = 1 ;
       }
}
function stop1(){
   clearInterval(IntervarID) ;
}
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*一定時間ごとに処理を繰り返す<BR>
 →アニメーションにスタート・ストップボタンを付ける(setInterval)<P>
<IMG SRC="image1.gif" NAME="animation" ALT="Animation" 
BORDER=0 WIDTH="100" HEIGHT="100">
<FORM>
   <INPUT TYPE="button" VALUE="スタート" onClick="Intervar
ID=setInterval('anime1()',500)">
   <INPUT TYPE="button" VALUE="ストップ" onClick="stop1()">
</FORM>
</BODY>
</HTML>
```

setTimeout()→「Windowオブジェクト」の「ステータス行に文字を流す」:P.292参照

JavaScript

カーソルの位置を取得する

オブジェクト名.captureEvents(Event.イベントタイプ | Event.イベントタイプ | ...)

[メソッド]

【サポートしているオブジェクト】
window･documentオブジェクト

Before

*カーソルの位置を取得する

X軸: 204
Y軸: 30

After

*カーソルの位置を取得する

X軸: 250
Y軸: 97

解説

「captureEvents()」メソッドは、取得したいイベントを設定することができます。複数のイベントを設定したい時は、「window.captureEvents(Event.イベントタイプ | Event.イベントタイプ | ...)」と記述します。
サンプルでは、「captureEvents()」で設定した時にのみ使用可能なイベントMouseMoveを設定して、マウスカーソルが移動するたびにカーソルの位置を取得し、フォームに書き出しています。
イベントタイプの記述は、サンプルのように「MOUSEMOVE」と記述してください。「MouseMove」というように小文字を混ぜて記述すると、Macintosh版のNetscape Navigator4.xでは正常に作動しますが、Windows版のNetscape Navigator4.xでは作動しません。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function Cap(e) {
    document.forms[0].elements[0].value = e.pageX;
    document.forms[0].elements[1].value = e.pageY;
    return true;
}
window.captureEvents(Event.MOUSEMOVE);
onMouseMove=Cap;
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*カーソルの位置を取得する<P>
<FORM NAME="ECAP">
X軸: <INPUT TYPE="text" NAME="ecap" SIZE=10>
<BR>
Y軸: <INPUT TYPE="text" NAME="ecap" SIZE=10>
</FORM>
</BODY>
</HTML>
```

onMouseMove→リファレンス「イベントタイプ」の「MouseMove」:P.551参照

イベントを引き渡す

オブジェクト名.handleEvent(Event.イベントタイプ) [メソッド]

【サポートしているオブジェクト(配列)
すべてのオブジェクト

Before

*イベントを引き渡す
Homeへ...

After

[JavaScript アプリケーション]
このページのリンクは現在機能しません!!
OK

[JavaScript アプリケーション]
申し訳ありませんm(_ _)m、もうしばらくお待ちください!!
OK

解説

「handleEvent()」メソッドは、取得したイベントを、他のオブジェクトに引き渡します。イベントを取り扱うすべてのオブジェクトで使用可能なメソッドです。
サンプルでは、関数「Cli()」の中で、「window.document.links[0].handleEvent(e)」によってリンク内に設定してしている「onClick="Cli2()"」にイベントを引き渡し、関数「Cli2()」を発生させています。
今の状態では、関数「Cli()」内で設定した警告用のダイアログボックスが開いた後、関数「Cli2()」で設定した警告用のダイアログボックスが開きます。「Cli()」内の「alert()」と「handleEvent()」の設定の順番を逆にすると、「Cli2()」の評価が「Cli()」内で設定した警告用のダイアログボックスが開く前に発生するので、ダイアログボックスが開く順番が逆になります。
「captureEvents()」メソッドでのイベントタイプの記述は、サンプルのように「CLICK」と記述してください。「Click」というたように小文字を混ぜて記述すると、Macintosh版のNetscape Navigator4.xでは正常に作動しますが、Windows版のNetscape Navigator4.xでは作動しません。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function Cli(e) {
    alert ("このページのリンクは現在機能¥しません!!");
    window.document.links[0].handleEvent(e);
    return false;
        }
function Cli2() {
    alert ("申¥し訳ありませんm(_ _)m、もうしばらくお待ちください!!");
    return true;
}
window.captureEvents(Event.CLICK);
onClick=Cli
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*イベントを引き渡す<P>
<A HREF="../../home.html" onClick="Cli2()">Homeへ...</A>
</BODY>
</HTML>
```

onClick→リファレンス「イベントタイプ」の「Click」:P.550参照

イベントを解放する

オブジェクト名.releaseEvents(Event.イベントタイプ)　[メソッド]

【サポートしているオブジェクト(配列)】
window・document・layerオブジェクト

Before

*イベントを解放する

home へ...(作成中)

After

[JavaScript アプリケーション]

申し訳ありませんm(_ _)m、現在リンクが切れています!!

OK

解説

「releaseEvents()」メソッドは、指定したイベントを解放します。
サンプルでは、リンクの中に「onMouseOut」のイベントを設定しているのですが、このイベントは、1度ウィンドウがクリックされて関数「Cli()」が発生し、関数内「releaseEvents(Event.MouseOut)」によってイベントが解放されるまでは評価されません。
つまり、リンクがクリックされるまでは、マウスカーソルがリンクの上に乗っても、外れても何も起こらないのです。ところが、1度リンクがクリックされると、それ以降はリンク上からマウスカーソルが外れるとイベントが発生し、警告用のダイアログボックスが開くようになります。
サンプルは、Macintosh版のNetscape Navigator4.xでのみ動作が確認されています。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function Cli(e) {
    window.releaseEvents(Event.MouseOut);
    return true;
}
function Mou() {
    alert ("申¥し訳ありませんm(_ _)m、現在リンクが切れています!!");
    return true;
}
window.captureEvents(Event.Click);
onClick=Cli
//-->
```

```
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*イベントを解放する<P>
<A HREF="http://" onMouseOut="Mou()">homeへ...(作成中)</A>
</BODY>
</HTML>
```

onClick→リファレンス「イベントタイプ」の「Click」:P.550参照
onMouseOut→リファレンス「イベントタイプ」の「Mouse Out」:P.551参照

注意

ソースをコピーする時の注意

JavaScriptは、ソースコードを直接HTMLファイルに書き込むため、比較的簡単にソースコードを見ることができます。これは、Internet上に数多く公開されているコピーフリーのJavaScriptのソースを、簡単に見ることができるということなので、JavaScriptを勉強する上で非常に有意義で、このことはJavaScriptの大きな魅力の1つでもあります。

しかし、Netscape Navigator3.0以上のバージョンのブラウザで、「表示」メニューの「ページのソース」を使用してソースコードを見ようとした場合、そこに表示されるソースコードは、正確ではない場合があります。

例えば、次のスクリプトを実行した時、

```
<SCRIPT language="JavaScript">
document.write("今日は!!")
</SCRIPT>
```

「ページのソース」で表示されるソースは、JavaScriptのコード部分がなくなり、次のようになる場合があります。

```
今日は!!
```

また、これと同じような例で、特殊フォントの「<」や「≶」も、「<」や「>」となります。

このように、どうやら「ページのソース」で表示されるソースは、HTMLファイルの内容そのままではなく、ブラウザに一旦読み込まれて評価された結果のHTMLのようです。

この問題を回避してJavaScriptのソースを正確に見るには、ブラウザの設定でJavaScriptの実行を一旦中止してからページを読み込み、ページのソースを表示するようにすれば大丈夫です。

同じ階層のイベントを取得する

オブジェクト名.**routeEvent(Event.**イベントタイプ) [メソッド]

【サポートしているオブジェクト】
window・document・layerオブジェクト

*同じ階層のイベントを取得する

[JavaScript アプリケーション]
documentはmouseupのイベントを取得しました。次にrouteEventからイベントを返します。
OK

[JavaScript アプリケーション]
routeEventからイベントを受け取りました。
OK

解説

「routeEvent()」メソッドは、「captureEvents()」で取得したのと同じイベントを返します。サンプルでは、「routeEvent(e)」が、マウスボタンが離された時に取得されたイベントと同等のイベントを返します。
JavaScript1.2で追加されたメソッドです。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
function fun1(e) {
   alert ("documentは" + e.type + "のイベントを取得しました。次に
routeEventからイベントを返します。");
   document.routeEvent(e);
   alert ("routeEventからイベントを受け取りました。");
   return true;
}
document.captureEvents(Event.onMouseUp);
document.onMouseUp=fun1;
//-->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*同じ階層のイベントを取得する<P>
</BODY></HTML>
```

onMouseUp→リファレンス「イベントタイプ」の「MouseUp」:P.552参照

数値かどうかを調べる

isNaN() ［ビルトイン関数］

*数値かどうかを調べる

false
true

解説

「isNaN()」メソッドは、あたえられた値が数値かどうかを調べます。
値が数値の場合は「false」を、数値でない場合は「true」を返します。
Netscape Navigator3.x以前のUNIX版以外のブラウザでは、機能しない場合があります。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*数値かどうかを調べる<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
var n = 1.23;
var m = "Mozi";
document.write(isNaN(n));
document.write("<BR>");
document.write(isNaN(m));
//--->
</SCRIPT>
</BODY>
</HTML>
```

NN4.06

IE5.0

有限数かどうかを調べる

isFinite()　　[ビルトイン関数]

*有限数かどうかを調べる

true
false
true

解説

「isFinite()」メソッドは、与えられた値が有限数かどうかを調べます。
もしも値が有限数の場合は「true」を、そうでない場合は「false」を返します。
サンプルでは、有限数である「1.23」と無限大を表す数値「Infinity」を、「isFinite()」使って評価しています。「!m」は、「Infinity」ではない、ということになるので「true」の値が返ります。
JavaScript1.3で追加されたビルトイン関数です。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*有限数かどうかを調べる<P>
<SCRIPT LANGUAGE="JavaScript1.3">
<!---
var n = 1.23;
var m = Infinity;
document.write(isFinite(n));
document.write("<BR>");
document.write(isFinite(m));
document.write("<BR>");
document.write(isFinite(!m));
//--->
</SCRIPT>
</BODY>
</HTML>
```

JavaScript

文字列を整数に変換する

parseInt() [ビルトイン関数]

NN2.0 NN3.0 NN4.0 NN4.06 IE3.0 IE4.0 IE5.0

*文字列を整数に変換する

1
0
1
NaN

解説

「parseInt()」メソッドは、文字列を整数に変換します。
小数点以下は切り捨てられ、数値でない場合は「NaN」を返します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*文字列を整数に変換する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
var n = "1.234"
var m = "0.777"
  document.write(parseInt(n))
  document.write("<BR>")
  document.write(parseInt(m))
  document.write("<BR>")
  document.write(parseInt(n)+parseInt(m))
  document.write("<BR>")
  document.write(parseFloat("文字といってもこれだと..."))
//--->
</SCRIPT>
</BODY>
</HTML>
```

NN2.0
NN3.0
NN4.0
NN4.06
IE3.0
IE4.0
IE5.0

文字列を浮動小数点数に変換する

parseFloat() ［ビルトイン関数］

*文字列を浮動小数点数に変換する

1.234
0.777
2.011
NaN

解説

「parseFloat()」メソッドは、文字列を浮動小数点数に変換します。
「parseInt()」メソッド(前項)と違い、少数点以下は切り捨てられません。数値でない場合は「NaN」を返します。

Sample

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*文字列を浮動小数点数に変換する<P>
<SCRIPT LANGUAGE="JavaScript">
<!---
var n = "1.234";
var m = "0.777";
  document.write(parseFloat(n));
  document.write("<BR>");
  document.write(parseFloat(m));
  document.write("<BR>");
  document.write(parseFloat(n)+parseFloat(m));
  document.write("<BR>");
  document.write(parseFloat("文字といってもこれだと..."));
//--->
</SCRIPT>
</BODY>
</HTML>
```

文字をASCII形式(URL形式)に変換する

escape()　　[ビルトイン関数]

Before

*文字をASCII形式(URL形式)に変換する

文字を変換する

変換!! Clear

Clear

After

*文字をASCII形式(URL形式)に変換する

文字を変換する

変換!! Clear

%u6587%u5B57%u3092%u5909%u63DB%u3059%u308B

Clear

解説

「escape()」メソッドは、文字をASCII形式の文字に変換します。
サンプルでは、上のフォームに文字を入力して「変換!!」ボタンを押すと、下のフォームにASCII形式に変換された文字が表示されます。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function dec1(C1) { document.Out1.Decode1.value = escape
(C1.Input1.value) }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*文字をASCII形式(URL形式)に変換する<P>
<FORM NAME="In1">
<TEXTAREA NAME="Input1" ROWS=5 COLS=50></TEXTAREA><P>
<INPUT TYPE="button" VALUE="変換!!" onClick="dec1(this.form)">
<INPUT TYPE=reset VALUE="Clear"><P>
</FORM>
<FORM NAME="Out1" >
<TEXTAREA NAME="Decode1" ROWS=5 COLS=50></TEXTAREA><P>
<INPUT TYPE=reset VALUE="Clear">
</FORM>
</BODY>
</HTML>
```

JavaScript

ASCII形式の文字をデコードする

unescape() [ビルトイン関数]

Before

*ASCII形式の文字をデコードする

%u6587%u5B57%u3092%u5909%u63DB%u3059%u308B

変換!! Clear

Clear

After

*ASCII形式の文字をデコードする

%u6587%u5B57%u3092%u5909%u63DB%u3059%u308B

変換!! Clear

文字を変換する

Clear

解説

「unescape()」メソッドは、ASCII形式の文字を文字に変換します。
サンプルでは、上のフォームにASCII形式の文字(フォームから送られてくる%付きの文字)を入力して「変換!!」ボタンを押すと、下のフォームに通常の文字に変換された文字が表示されます。

Sample

```
<HTML><HEAD><TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function dec(C) { document.Out2.Decode2.value = unescape
(C.Input2.value) }
//--->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
*ASCII形式の文字をデコードする<P>
<FORM NAME="In2">
<TEXTAREA NAME="Input2" ROWS=5 COLS=50></TEXTAREA><P>
<INPUT TYPE="button" VALUE="変換!!" onClick="dec(this.form)">
<INPUT TYPE="reset" VALUE="Clear">
<P>
</FORM>
<FORM NAME="Out2" >
<TEXTAREA NAME="Decode2" ROWS=5 COLS=50></TEXTAREA><P>
<INPUT TYPE=reset VALUE="Clear">
</FORM></BODY></HTML>
```

その他のビルトイン関数

taint()　【ビルトイン関数】

解説

JavaScriptでは、セキュリティのため、通常は別のウィンドウやフレームからプロパティなどの値を参照できないようになっています。「taint()」メソッドは、一時的にその規制を無効にします。
参照可能にできるのは、プロパティ・関数・オブジェクトなどの値です。また、「taint()」メソッドが有効な状態かどうかを調べるためには、navigatorオブジェクトの「taintEnabled()」メソッドを使用します。
「taint()」メソッドが有効な時のみ使用可能なプロパティとしては、documentオブジェクトの「domain」プロパティ・historyオブジェクトの「current」プロパティ・「next」プロパティ・「previous」プロパティがあります。
JavaScript1.1で追加されたビルトイン関数ですが、Netscape Navigator4.0からはセキュリティ対策にSigned Scriptを使うことになったため、JavaScript 1.2では削除されました。

untaint()　【ビルトイン関数】

解説

「untaint()」メソッドは、「taint()」メソッドによって一時的に別のウィンドウやフレームから参照可能になったプロパティなどの値を、再び参照できない状態に戻します。
Netscape Navigator3.0(JavaScript1.1)で追加されたビルトイン関数ですが、Netscape Navigator4.0からはセキュリティ対策にSigned Scriptを使うことになったため、JavaScript1.2では削除されました。

注意

"taint()"及び"untaint()"は、通常のブラウザでは機能しません。

1. JavaScriptで取り扱える型の種類

JavaScript内で取り扱える値のタイプは、次の通りです。

文字列

ダブルクォーテーションマーク「"」、またはシングルクォーテーションマーク「'」で囲まれた文字や、数字。

【例】`"123" , "Moziretu" , '文字列'`

以下の特殊記号も文字列です。

¥b	バックスペース
¥f	フォームフィード
¥n	改行
¥r	キャリッジリターン
¥t	タブ
¥¥	バックスラッシュ

数値

・整数

8進数、10進数 、16進数が使えます。

8進数は0から始まり0から7までで表す数値のことを、10進数は0以外から始まり0から9までで表す数値のことをいいます。16進数は0xから始まり0から9までの数値と、それに続くaからf、あるいはAからFまでのアルファベットで表す数値のことをいいます。

【例】`0514 , 156 , 0x11`

・浮動小数点数

小数点をピリオド(.)で表わした10進数、又はeあるいはEを使用する指数が使えます。

【例】`3.14 , 1.79E+308`

論理値

値を比較した時、その比較が論理的に考えて正しいかどうかの値。

正しければ「真」となり、正しくない場合は「偽」となります。例えば、「1==1」は正しいので「真」となり、「1==2」は正しくないので「偽」となります(「==」に関しては、右ページを参照)。

true	真の値
false	偽の値

null値

プロパティなどに値が定義されていなかったり、何も設定されていない状態を表します。

null	未定義、何も設定されていない状態

2. 演算子

JavaScriptで使用できる値の計算や、比較などに用いる記号は、以下の通りです。

算術演算子

算術演算をするための演算子。

=	変数に値を代入する
+	加算
-	減算又は負の値を表す
*	乗算
/	除算
%	乗除、除算で余りを求める
++	値に1を増やす(インクリメント)
--	値から1を引く(デクリメント)

インクリメント、デクリメントの用法は次の通りです。

y = x++	yに値を代入してからxに1を加える
y = x--	yに値を代入してからxから1を引く
y = ++x	xに1加えてからyに値を代入する
y = --x	xから1引いてからyに値を代入する

比較演算子

左辺と右辺の値を比較して、真の時は「true」の、偽の時は「false」の値を返します。

x == y	xはyと等しい
x != y	xとyとは等しくない
x < y	xはyより小さい
x <= y	xはyより小さいか等しい
x > y	xはyより大きい
x >= y	xはyより大きいか等しい

(JavaScript1.3での追加分)

x === y	xはyと等しい。形の変更を行わない
x !== y	xとyとは等しくない。 形の変更を行わない

論理演算子

左辺と右辺を論理演算し、真の時は「true」の、偽の時は「false」の値を返します。

x && y	AND(xかつy)
x ll y	OR(xまたはy)
x ! y	NOT(xはyでない)

代入演算子

右辺の値を左辺に代入する演算子。

x += y	左辺に右辺の値を加算し結果を左辺に代入(x=x+yと同じ)
x -= y	左辺に右辺の値を減算し結果を左辺に代入(x=x-yと同じ)
x *= y	左辺に右辺の値を乗算し結果を左辺に代入(x=x*yと同じ)
x /= y	左辺に右辺の値を除算し結果を左辺に代入(x=x/yと同じ)
x %= y	左辺に右辺の値を 除算し余りを左辺に代入(x=x%yと同じ)

文字列演算子

文字列の連結を行う演算子。

"文字列A" + "文字列B"	「文字列A」と「文字列B」を連結する
a += "文字列B"	aの後に「文字列」を追加する

ビット演算子

値をビット単位で演算します。

コンピュータは、文字列も数値もすべてビット単位(2進数)で処理しています。ビット演算子は、値をコンピュータと同じように、ビット単位で取り扱う演算子です。

~	ビットの反転
&	ビットの論理積(AND)
|	ビットの論理和(OR)
^	ビットの排他的和(XOR)
<<	ビットの左シフト
>>	ビットの右シフト
>>>	ビットの論理右シフト
<<=	ビットごとの左シフトの代入
>>=	ビットごとの右シフトの代入
>>>=	論理右シフトの代入

new演算子

オブジェクトのインスタンスを作成を行う演算子。

```
オブジェクト名 = new オブジェクトタイプ(値1,値2,...値n)
```

条件演算子

条件式が真(true)の時とfalse(偽)の時で、違った処理を行う演算子。

条件式？x:y	条件式を評価して、真(true)の時はxの、false(偽)の時はyの値を参照する

カッコ、その他

式や値を括るカッコや、オブジェクトやメソッドを区切るピリオドなども演算子です。

[]	配列の添番を括る
()	値や数式を括る
.	オブジェクト・プロパティ・メソッドを区切る

typeof()演算子(JavaScript1.1)

数値・文字列・変数・オブジェクトなどの型(タイプ)を調べる演算子。

【例】

```
var suuzi = 123
var mozi = "今日は"
var kye = null
var today = new Date()
var kan = blur
document.write(typeof(suuzi) + "<BR>")
document.write(typeof(mozi) + "<BR>")
document.write(typeof(kye) + "<BR>")
document.write(typeof(today) + "<BR>")
document.write(typeof(kan) + "<BR>")
document.write(typeof(muimi))
```

void()演算子(JavaScript1.1)

値を返さずに式を評価する演算子。

【例】

```
<A HREF="javascript:void(Kansuu())">
この文字をクリックすると Kansuu() が発生します
</A>
```

delete演算子(JavaScript1.2)

指定したオブジェクト・プロパティ・配列の要素の削除を行う演算子。

「delete」の処理が正常に行われた場合、「undefined」と「true」の値を返し、そうでなければ「false」の値を返します。

```
delete オブジェクト名
delete オブジェクト名.プロパティ名
delete 配列名[インデックス]
```

サンプルでは、「delete」演算子を使って「hairetu[2]」の配列の要素を削除しているので、「document.write()」で書き出された結果は、「0番目/1番目//3番目」となります。

【例】

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
hairetu= new Array("0番目","1番目","2番目","3番目");
delete hairetu[2];
document.write(hairetu.join("/"));
//--->
</SCRIPT>
```

演算子の優先順位

それぞれの演算子には、優先順位があります。
上に行くほどに優先順位が高くなります。

```
() [] .
! ~ - ++ -- typeof void
* / %
+ -
<< >> >>>
< <= > >=
== !=
&
^
|
&&
||
?:
= += -= *= /= %= <<= >>= >>>= &= ^= |=
```

JavaScript

リファレンス

注意 文字列型と数値型の取り扱いについて

JavaScript の文字列型と数値型の区別は、非常に曖昧です。

例えば次のスクリプト場合、本当ならば "2" は文字列型で、2 は数値型なので「false」になるはずなのですが、JavaScript1.2 以外では「true」の値を返します。

```
<SCRIPT LANGUAGE="JavaScript">
<!--
document.write("2" == 2)
//-->
</SCRIPT>
```

この時に注意が必要なのは、JavaScript1.2 では、文字列型と数値型を明確に区別することです。

例えば、次のスクリプト場合は「false」の値が返ります。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("2" == 2)
//-->
</SCRIPT>
```

しかし、JavaScript1.2でも次の例のように文字列に-0を設定すると、その他のバージョンのJavaScriptと同じように「true」の値を返すようになります。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write(("2"-0) == 2)
//-->
</SCRIPT>
```

このように、JavaScript1.2 で明確に区別されるようになった文字列型と数値型なのですが、JavaScript1.3になって再び、それ以前のバージョンのJavaScriptのように再び区別されないようになりました。

例えば次のスクリプト場合、JavaScript1.2 以前のバージョンと同様に「true」の値が返ります。

```
<SCRIPT LANGUAGE="JavaScript1.3">
<!--
document.write("2" == 2)
//-->
</SCRIPT>
```

これは、ECMAScript と仕様をあわせるためです。

JavaScript1.3 で、JavaScript1.2 のように文字列型と数値型を区別して値を比較したい場合は、JavaScript1.3 で新たに追加された演算子 "===" や "!==" を使用します。

例えば、次のように値の比較に "===" を使用すると、JavaScript1.2 のように文字列型と数値型を明確に区別し、「false」の値が返ります。

```
<SCRIPT LANGUAGE="JavaScript1.3">
<!--
document.write("2" === 2)
//-->
</SCRIPT>
```

3. JavaScriptの命令文(ステートメント)

JavaScript1.0 及び JavaScript1.1 で使用可能な命令文

ループから抜ける(break)

"for" や "while" などで、繰り返し(ループ)処理を行っている時に使用します。

「条件式」が真(true)の時、1番内側のループを抜けます。

【構文】

```
brerk;
```

【記述例】

```
for (i=1; i<=10; i++) {
    document.write("この文章を10回書き出します。:" +i+ "回目");
        if (i==5)
        break;
    document.write("でも...。<BR>");
}
document.write("<BR>break があるので5回でループを抜けます。<BR>");
```

コメント

JavaScript内にコメントを書く時に使用します。

"//"は、それ以降の1行が、"/*　　*/"は、間に挟まれた文字列が、コメントとして扱われます。

【構文】

```
//
/*    */
```

【記述例】

```
//この1行はコメントとなります。

/*これに囲まれている部分
  はコメントとなります。*/
```

処理のスキップ(continue)

"for" や "while" などで、繰り返し(ループ)処理を行っている時に使用します。

「条件式」が真(true)の時、"continue"以下の処理を飛ばして、繰り返し処理を続けます。

【構文】

```
continue;
```

【記述例】

```
for (i=1; i<=10; i++) {
        if (i==5)
        continue;
    document.write("この文章を10回書き出します:" +i+ "回目でも...。<BR>");
}
document.write("<BR>continueがあるので5回目がありません<BR>");
```

繰り返し処理 (for文)

「条件式」が真(true)の間、「処理」を繰り返し行います。

【構文】

```
for (初期値; 条件式; 増減式 ) { 処理 }
```

【記述例】

```
for (i=1; i<=4; i++) {
    document.write("この文章を4回書き出します。:" +i+ "回目<BR>");
}
```

重要 コメント使用時の注意

JavaScript 内で「<!--HTML のコメント -->」を使っても、エラーにはなりません。

しかし、JavaScript を無効にしている、あるいは未対応のブラウザで見ると、「<!--HTML のコメント -->」以下のソースコードが隠されずに丸見えになってしまいます。

プロパティ・メソッドの一覧(for...in文)

「オブジェクト」内のプロパティ・メソッドを「変数」に代入しながら順番に取り出します。

【構文】

```
for (変数 in オブジェクト) { 処理 }
```

【記述例】

```
for (var n in navigator){
      document.write(n,"<BR>");
}
```

関数の設定(function)

JavaScriptでは、一定の処理手続きを「関数」として設定することができます。

「function」の後に「関数名」とその「処理」を記述し、ページが読み込まれたタイミングや、イベントハンドラによってイベントが発生したタイミングで、「関数名」を呼ぶことによって「処理」が実行されます。

通常、関数の設定は、HTMLファイルの「<HEAD></HEAD>」タグ内で行います。これは、関数が設定される前に、「<BODY></BODY>」内で設定する「関数名」が呼ばれるのを防ぐためです。

【構文】

```
function 関数名([引数] [, 引数] [...,
引数]) { 処理 }
```

【記述例】

```
function kansuu(x,y) {
    document.write( x +"たす" + y
+ "は" + eval(x+y) +"です。");
}
kansuu(1,2)
```

条件分岐(if文)

「条件式」が真(true)の時は「処理1」を行い、それ以外の時は「処理2」を行います。

「else」の処理がない時は省略可能です。

【構文】

```
if (条件式) { 処理1 }
      else{ 処理2 }
```

【記述例】

```
var now = new Date();
var AMPM = now.getHours();
    if (AMPM < 12 ){ document.
write("午前") }
       else { document.write("午後
") }
```

値の代入(var)

変数、配列、オブジェクトなどの宣言を行います。

「= 値」を設定すれば、「var 名」にその値が代入されます。

【構文】

```
var var名 [= 値] [..., var名 [= 値] ]
```

【記述例】

```
var ataiA =5;
var ataiB =3;
var ataiC ="たす";
document.write( ataiA + ataiC +
ataiB + "は" + eval(ataiA+ataiB) +
"です。");
```

繰り返し処理(while文)

「条件式」が真(true)の間、「処理」を繰り返し行います。

【構文】

```
while( 条件式 ) { 処理 }
```

【記述例】

```
var i = 1
while ( i <= 4 ) {
    document.write("この文章を4回書き出
します。:" +i+ "回目<BR>");
     i++;
}
```

オブジェクトの省略(with文)

「オブジェクト」で指定したオブジェクトを省略して、プロパティやメソッドを使用できるようにします。

【構文】

```
with (オブジェクト) { 処理 }
```

【記述例】

```
with (document) {
    write("この文章は、documentの<BR>");
    write("部分を省略しています。<BR>");
}
```

戻り値を返す(return)

命令文内で値を返す時に使用します。
返す値がない時は不要です。

【構文】

```
return 値
```

【記述例】

```
<A HREF="../../home.html"onMouseOver="window.status='ステータス行にメッセージが出ます';return true">この文字の上にマウスカーソルを持ってくると</A>
```

オブジェクトの参照(this)

this(キーワード)の後に、指定したオブジェクトを参照します。

オブジェクトは、継承関係を気にすることなく直接指定できます。

【構文】

```
this.オブジェクト
```

【記述例】

```
<SCRIPT LANGUAGE="JavaScript">
<!---
function  Namae(n) { alert("入力された名前は"+n+"はです!!") }
//--->
</SCRIPT>
名前を入力して下さい<BR>
<FORM NAME="NAMAE">
<INPUT TYPE="Text" NAME="namae" onBlur="Namae(this.value)" value="">
</FORM>
```

オブジェクトを作成する(インスタンスの作成)(new)

ユーザー独自のオブジェクトを作成したり、ビルトインオブジェクトを使用する時は、new 演算子を使って、「オブジェクト名」で指定した名前を持った新たなオブジェクトを作成します。

【構文】

```
オブジェクト名 = new オブジェクトの型(値)
```

【記述例】

```
<SCRIPT LANGUAGE="JavaScript">
<!---
NewDay = new Date("may 2, 1997 23:00:00");
document.write(NewDay);
document.write("<BR>");
NewDay.setDate(23);
document.write(NewDay);
//--->
</SCRIPT>
```

配列の作成

"function MakeArray(n) {"から"return this ;}"までの処理で配列の要素を入れる器を作り、"要素名 = new MakeArray(要素数) ;"以下で配列の要素を流し込み、配列を作成します。

作成された配列は、"要素名[添番]"で希望の要素を取り出すことができます。

Netscape Navigator3.0(JavaScript1.1)からは、配列を扱うArrayオブジェクトが追加され、もっと簡単に配列を作成できるようになりました。また、Arrayオブジェクトの一部の機能は、Netscape Navigator2.0でも使用可能です。詳しくは、「Arrayオブジェクト」の「曜日を表示する(arrayオブジェクトを使う)」(P.494) を参照してください。

【記述例】は、「Date オブジェクト」の「曜日を表示する」(P.434) のサンプルを配列を使って書き直したものです。

【構文】

```
function MakeArray(n) {
        this.length = n ;
        for (var i = 0; i <= n; i++) {
                this[i] = 0;
        }
        return this ;
}
要素名 = new MakeArray(要素数) ;
要素名[添番] = 要素 ;
        .
        .
        .
```

【記述例】

```
<HTML>
<HEAD>
<TITLE>ARRAY_SAMPLE_1</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function MakeArray(n) {
    this.length = n ;
    for (var i = 0; i <= n; i++) {
        this[i] = 0;
    }
```

```
   return this ;
}
youbi = new MakeArray(7) ;
youbi[0] = "日";
youbi[1] = "月";
youbi[2] = "火";
youbi[3] = "水";
youbi[4] = "木";
youbi[5] = "金";
youbi[6] = "土";
function gety(y){ document.write(
youbi[y] ) }
//--->
</SCRIPT>
</HEAD>
<BODY>
*配列を使った曜日表示<P>
(
<SCRIPT LANGUAGE="JavaScript">
<!---
day = new Date();
gety(day.getDay());
//--->
</SCRIPT>
)
</BODY>
</HTML>
```

HTMLを配列として扱う

リンクやフォームなどのHTMLのタグは、JavaScriptのオブジェクトとして取り扱えるのと同時に、配列の要素としても取り扱えます。

HTMLファイルに記述されている、それぞれのオブジェクトの上から、添番が[0].[1].[2]...となる配列の要素が作成されます。それらの要素は"オブジェクトの配列名[添番]"で取り扱うことができます。

【構文】

オブジェクトの配列名[添番]

【オブジェクトの配列名】

anchors,applets,elements,embeds,forms,frames,history,images,layers,links,mineTypes,options,plugins
(各配列には、配列の数を持ったlengthプロパティがあります)

【記述例】

```
<HTML>
<HEAD>
<TITLE>ARRAY_SAMPLE_2</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function arr() {
    document.forms[0].elements[0].
value = "formのelements配列[0]の要素";
    document.forms[0].elements[1].
value = "formのelements配列[1]の要素";
    document.forms[0].elements[2].
value = "formのelements配列[2]の要素";
    document.forms[0].elements[3].
value = "formのelements配列[3]の要素";
}
//---->
</SCRIPT>
</HEAD>
<BODY onLoad="arr()">
*HTMLを配列として捉う<P>
<FORM NAME="ARRAY" >
<INPUT TYPE="text" NAME="array1"
 SIZE=40>
<BR>
<INPUT TYPE="text" NAME="array1"
 SIZE=40>
<BR>
<INPUT TYPE="text" NAME="array1"
 SIZE=40>
<BR>
<INPUT TYPE="text" NAME="array1"
 SIZE=40>
</FORM>
</BODY>
</HTML>
```

JavaScript1.2で追加された演算子

ラベル

式にラベルを設定しておくことにより、if文やfor文、while文のようなループ文から「break」や「continue」で抜ける時、ラベル名を指定して明示的に抜ける場所を設定することができます。

【記述例】では、まず変数「i」に「4」の値を、変数「j」に「2」の値を代入しています。そして、ラベル名で「checkiandj」と名前を付けたif文の処理と、ラベル名で「checkj」と名前を付けたif文の処理を用意しています。もし、変数「k」の値が「1」の時は、条件式「k==1」が真(true)となり、「break checkj」で「checkj」の処理を抜けるため、その下の「i」と変数「j」の合計を出す処理を行なわずに、

「i-j」が実行されます。「k」の値が「1」以外の時は、「i」と「j」の合計を出した後「break checkiandj」で「checkiandj」の処理を抜けるため、「i-j」の処理は実行されません。

【構文】

```
ラベル名 :
       式
break ラベル名
continue ラベル名
```

【記述例】

```
var i=4;
var j=2;
checkiandj :
if (4==i) {
 document.write("kの値は" + k + ".<br>");
  document.write("iの値は" + i + ".<br>");
  checkj :
   if (2==j) {
    document.write("jの値は" + j + ".<br>");
     if (1==k) { break checkj }
    document.write("iとjの合計" + (i+j) + ".<br>");
    break checkiandj ;
  }
   document.write(i + "-" + j + "=" + (i-j) + ".<br>");
  }
```

繰り返し処理(do while文)

do while文は、条件式がfalse(偽)になるまで、処理を繰り返し行なうような場合に使用します。

【記述例】では、"i<10"がfalse(偽)になるまでiに1を加えながら書き出しています。

【構文】

```
do 処理
     while (条件式);
```

【記述例】

```
var i=0;
do {
      i+=1;
      document.write(i,"<BR>");
      } while (i<10);
```

スイッチ(switch文)

switch文は、処理にラベルを設定し、各ラベルと値を参照し、真(true)になる処理を実行します。この時、もし真(true)になるラベルがない場合は、"default:"の処理を実行します。

【記述例】では、変数「i」の値が「Oranges」の時には「オレンジは1個100円です。」という文字が、「i」の値が「Apples」の時は「りんごは1個150円です。」という文字が、documentオブジェクトのwrite()メソッドによって書き出されます。また、「i」の値が「Oranges」でも「Apples」でもない場合は、「default:」で設定した、13行目の「申し訳ありません。ただいま品切れです。」という文字が書き出されます。

【構文】

```
switch (値) {
      case ラベルA :
            処理;
            break;
      case ラベルB :
            処理;
            break;
            .
            .
            .
      default:
            処理;
      }
```

【記述例】

```
switch (i) {
      case "Oranges" :
            document.write("オレンジは1個100円です。<BR>");
            break;
      case "Apples" :
            document.write("りんごは1個150円です。<BR>");
            break;
      default:
            document.write("申し訳ありません。ただいま品切れです。");
}
document.write("なお、上記金額に消費税は含まれておりません。<BR>");
```

4. イベントハンドラ

ユーザーやスクリプトによってページがロードされたり、オブジェクトがクリックされたりというような、特定の動作が起こったタイミングをイベントといいます。JavaScriptでは、イベントの発生を取得し、そのタイミングでスクリプトの実行を開始することができます。このイベントの取得を行う指定を、イベントハンドラといいます。

イベントハンドラの設定は、そのイベントハンドラを設定可能なオブジェクトの、HTMLタグ内に記述することによって行います。

JavaScriptで用意されているイベントハンドラと、そのイベントハンドラがどのようなイベントを取得し、どのオブジェクトに対応しているかは、次の通りです。

onAbort

(JavaScript1.1～)

画像の読み込みがキャンセルされた時のイベントを取得するイベントハンドラ。

画像が読み込まれている途中に、ブラウザのストップボタンを押すなどの動作で、画像の読み込みがキャンセルされた時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】

(JavaScript1.0)
未対応

(JavaScript1.1)
Imageオブジェクト

onBlur

(JavaScript1.0～)

フォーカスが、フォームやウィンドウから離れた時のイベントを取得するイベントハンドラ。

Textフォームなどでマウスカーソルが点滅していたり、RadioフォームやCheckboxフォームがチェックされていたり、ウィンドウがアクティブになっているなどの状態が、フォーカスがある状態になります。

フォームを移動したり、フォーカスがあるウィンドウとは別ウィンドウをクリックすることなどで、フォーカスが移動した時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】

(JavaScript1.0)
Selectオブジェクト
Textオブジェクト
Textareaオブジェクト

(JavaScript1.1)
Buttonオブジェクト
Checkboxオブジェクト
FileUploadオブジェクト
frameオブジェクト
Passwordオブジェクト
Radioオブジェクト
Resetオブジェクト
Submitオブジェクト
windowオブジェクト

onChange

(JavaScript1.0～)

フォームの内容に変化があり、フォーカスがフォームから離れた時のイベントを取得するイベントハンドラ。

Text フォームの内容を変更し、他のフォームに移動することなどで、フォーカスが移動した時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】

(JavaScript1.0)
Selectオブジェクト
Textオブジェクト
Textareaオブジェクト

(JavaScript1.1)
FileUploadオブジェクト

onClick

(JavaScript1.0)

ボタンやリンクをクリックした時のイベントを取得するイベントハンドラ。

Buttonフォームやリンクをクリックした時をイベントとして取得し、設定した処理を実行します。

JavaScript1.1から、"return false"と"false"を返すと、リンクの参照などの通常動作を中止できるようになりました。これにより、リンクにこのイベントハンドラを設定して、JavaScriptの処理の一番最後に"return false"を返すように設定することで、リンクをクリックしてもJavaScriptのみを実行し、別ページに移動するなどのリンクの処理を行わないようにすることができます。

【利用可能なオブジェクト】
(JavaScript1.0)
Buttonオブジェクト
Checkboxオブジェクト
Linkオブジェクト
Radioオブジェクト
Resetオブジェクト
Submitオブジェクト

(JavaScript1.1)
オブジェクトの追加なし

onError

(JavaScript1.1～)
ページや画像の読み込エラーが発生した時のイベントを取得するイベントハンドラ。
画像読み込み時に、リンク切れなどで画像の読み込みがうまく行われなかった時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】
(JavaScript1.0)
未対応

(JavaScript1.1)
Imageオブジェクト
windowオブジェクト

onFocus

(JavaScript1.0～)
フォーカスがフォームやウィンドウにあたえられた時のイベントを取得するイベントハンドラ。
フォームを移動したり、フォーカスがないウィンドをクリックすることなどで、フォーカスが移動した時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】
(JavaScript1.0)
Selectオブジェクト
Textオブジェクト
Textareaオブジェクト

(JavaScript1.1)
Buttonオブジェクト
Checkboxオブジェクト
FileUploadオブジェクト
frameオブジェクト
Passwordオブジェクト
Radioオブジェクト
Resetオブジェクト
Submitオブジェクト
windowオブジェクト

onLoad

(JavaScript1.0～)
ページや画像が読み込まれた時のイベントを取得するイベントハンドラ。
ページや画像の読み込みが終了した時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】
(JavaScript1.0)
windowオブジェクト

(JavaScript1.1)
Imageオブジェクト

onMouseOut

(JavaScript1.1～)
マウスが指定された領域から離れた時のイベントを取得するイベントハンドラ。
リンクに設定することによって、マウスカーソルがリンクから離れた時をイベントとして取得し、設定した処理を実行することができます。

【利用可能なオブジェクト】
(JavaScript1.0)
未対応

(JavaScript1.1)
Linkオブジェクト
Areaオブジェクト

onMouseOver

(JavaScript1.0〜)

マウスが指定された領域内に入った時のイベントを取得するイベントハンドラ。

リンクに設定することによって、マウスカーソルがリンク上に乗った時をイベントとして取得し、設定した処理を実行することができます。

【利用可能なオブジェクト】

(JavaScript1.0)
Linkオブジェクト

(JavaScript1.1)
Areaオブジェクト

onReset

(JavaScript1.1〜)

フォームがリセットされた時のイベントを取得するイベントハンドラ。

Resetフォームが押されるなどで、フォームの内容がリセットされた時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】

(JavaScript1.0)
未対応

(JavaScript1.1)
Formオブジェクト

onSelect

(JavaScript1.0〜)

フォームのテキスト領域が選択された時のイベントを取得するイベントハンドラ。

Text フォームや Textarea フォームが選択され、フォームへの入力が可能になった時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】

(JavaScript1.0)
Textオブジェクト
Textareaオブジェクト

(JavaScript1.1)
オブジェクトの追加なし

onSubmit

(JavaScript1.0〜)

フォームのSubmitボタンが押されたときた時のイベントを取得するイベントハンドラ。

Submitボタンが押され、フォームのデータ送信が開始される時をイベントとして取得し、設定した処理を実行します。

データ送信の処理はJavaScriptの処理が終了するまで行われず、JavaScriptの処理で"return false"と" false"を返すと、データ送信の処理は中止されます。

【利用可能なオブジェクト】

(JavaScript1.0)
Formオブジェクト

(JavaScript1.1)
オブジェクトの追加なし

onUnload

(JavaScript1.0〜)

別のページに移動した時のイベントを取得するイベントハンドラ。

今のページを抜けて、別ページに移動した時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】

(JavaScript1.0)
windowオブジェクト

(JavaScript1.1)
オブジェクトの追加なし

5. イベントタイプ

JavaScript1.2からは、イベントをオブジェクトとして捕らえるeventオブジェクトが追加されました。そのため、イベントを取得したいオブジェクトに対して取得するイベントのタイプを設定することによって、そのオブジェクト上のどこでもイベントの発生を取得することができるようになりました。

JavaScriptで設定できるイベントタイプと、そのイベントタイプが設定できるオブジェクト、イベントタイプで取得できる値、つまりeventオブジェクトのプロパティは、次の通りです。

Click

マウスがクリックされた時のイベントを取得します。

"MouseDown"イベントと"MouseUp"イベントを合わせたもの。

【利用可能なオブジェクト】

Button, Checkbox, Link, Radio, Reset, Submit オブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"Click"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ
which	左ボタンの時は"1"を、右ボタンの時は"3"を値に持つ
modifiers	修飾キーの値を持つ

button がクリックされても、layerX,layerY, pageX, pageY,screenX,screenYは値を返しません。

DblClick

マウスがダブルクリックされた時のイベントを取得します。

Windows版とMacintosh版で使用可能です。

【利用可能なオブジェクト】

document, Area, Linkオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	" DblClick"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ
which	左ボタンの時は"1"を、右ボタンの時は"3"を値に持つ
modifiers	修飾キーの値を持つ

DragDrop

ウィンドウ上にファイルやショートカットなどをドラッグアンドドロップした時のイベントを取得します。

イベントが発生した時にtrue(真)を返せばドラッグアンドドロップを許し、false(偽)を返せばドラッグアンドドロップの動作を中止します。

Windows版のNetscape Navigator4.xは、画像のドラッグアンドドロップに対応していません。

【利用可能なオブジェクト】

windowオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"DragDrop"を値に持つ
data	ドロップされたファイル等のURLを返す

KeyDown

ユーザーがキーを押した時のイベントを取得します。

"KeyDown"イベントは"KeyPress"イベントより前に発生し、もしも"KeyDown"イベントがfalse(偽)を返した時は、"KeyPress"イベントは発生しません。

【利用可能なオブジェクト】

document, Image, Link,Textareaオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"KeyDown"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ
which	押されたキーのASCIIの値を持つ
modifiers	修飾キーの値を持つ

KeyPress

ユーザーがキーを押たままの状態にした時のイベントを取得します。

"KeyDown"イベントがtrue(真)の値を返した時のみイベントが発生し、ユーザーがキーを放すまでイベントは発生し続けます。

【利用可能なオブジェクト】

document, Image, Link,Textareaオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"KeyPress"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ
which	押されたキーのASCIIの値を持つ
modifiers	修飾キーの値を持つ

KeyUp

ユーザーがキーを放した時のイベントを取得します。

【利用可能なオブジェクト】

document, Image, Link,Textareaオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"KeyUp"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ
which	放されたキーのASCIIの値を持つ
modifiers	修飾キーの値を持つ

MouseDown

ユーザーがマウスボタンを押した時のイベントを取得します。

もしも"MouseDown"イベントがfalse(偽)を返した時には、選択やリンクの参照などの通常動作が中止されます。

【利用可能なオブジェクト】

Button, document,Linkオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"MouseDown"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ
which	左ボタンの時は"1"を、右ボタンの時は"3"を値に持つ
modifiers	イベントが発生した時に押された修飾キーの値を持つ

MouseMove

カーソルが動いた時のイベントを取得します。

"captureEvents()"メソッドでこのイベントを取得するように設定している時のみ、このイベントの取得を有効にすることができます。

【利用可能なオブジェクト】

なし

【eventオブジェクトのプロパティで使われた時】

type	"MouseMove"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ

MouseOut

オブジェクトからマウスカーソルが離れた時のイベントを取得します。

【利用可能なオブジェクト】

Area, layer,Linkオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"MouseOut"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ

MouseOver

オブジェクトにカーソルが乗った時のイベントを取得します。

【利用可能なオブジェクト】

Area, layer,Linkオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"MouseOver"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ

MouseUp

ユーザーがマウスボタンを放した時のイベントを取得します。

もしも"MouseUp"イベントがfalse(偽)を返した時には、選択やリンクの参照などの通常動作が中止されます。

【利用可能なオブジェクト】

Button, document,Linkオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"MouseUp"を値に持つ
layerX, layerY, pageX, pageY, screenX,screenY	イベントが発生した時のカーソルのX軸Y軸位置を値に持つ
which	左ボタンの時は"1"を、右ボタンの時は"3"を値に持つ
modifiers	イベントが発生した時に押された修飾キーの値を持つ

Move

ユーザー、又はスクリプトによって、ウィンドウ、又はフレームが動いた時のイベントを取得します。

【利用可能なオブジェクト】

window,frameオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"Move"を値に持つ
screenX,screenY	ウィンドウ、又はフレームの左上角の位置の値を持つ

Resize

ユーザー、又はスクリプトによって、ウィンドウ、又はフレームのサイズが変更された時のイベントを取得する。

【利用可能なオブジェクト】

window,frameオブジェクト

【eventオブジェクトのプロパティで使われた時】

type	"Resize"を値に持つ
width,height	ウィンドウ、又はフレームの幅と高さの値を持つ

6. ナビゲータオブジェクト

navigatorオブジェクト

ブラウザのユーザーエージェントやアプリケーション名、バージョンなど、ブラウザ固有の情報を提供するオブジェクト。

通常、サーバー側で行われているユーザーエージェントなどからブラウザを判断し、そのブラウザに最適化されたWebページを表示させるなどの処理を、ブラウザ側で行うことを可能にします。

【用法】

```
navigator.property
navigator.method()
```

【プロパティ】

(JavaScript1.0)

appCodeName	ブラウザのコード名
appName	ブラウザのブラウザ名
appVersion	ブラウザのバージョン
userAgent	ブラウザのユーザーエージェント

(JavaScript1.1)

mimeTypes	MIMEタイプ配列(オブジェクト)
plugins	プラグイン配列(オブジェクト)

(JavaScript1.2)

language	ユーザーが選択した言語
platform	ユーザーのプラットホームのタイプ

【メソッド】

(JavaScript1.0)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

javaEnabled()	Javaが使えるか
taintEnabled()	外部からのプロパティ参照を許可しているかどうか
valueOf()	オブジェクトの値を返す

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

mimeTypesオブジェクト(配列) (JavaScript1.1〜)

ブラウザで使用できる使用MIMEタイプの配列を作成する、navigatorオブジェクトのプロパティ。

【用法】

```
navigator.mimeTypes[インデックス].pro
perty
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

description	MIMEタイプの詳細
enablePlugin	プラグインを使うための名前
length	MIMEタイプの数
suffixes	MIMEタイプの拡張子
type	MIMEタイプ名

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

なし

【イベントハンドラ】

なし

pluginsオブジェクト(配列) (JavaScript1.1〜)

ブラウザにインストールされている使用可能なプラグインの配列を作成する、navigatorオブジェクトのプロパティ。

【用法】

```
navigator.plugins[インデックス].prope
rty
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

description	プラグインの詳細
filename	プラグインのファイル名
length	プラグインの数
name	プラグインの名前

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

refresh()	プラグイン配列のリフレッシュ

【イベントハンドラ】

なし

screenオブジェクト(JavaScript1.2~)

ディスプレイに関する情報を提供するオブジェクト。

【用法】

```
screen.property
```

【プロパティ】

(JavaScript1.2)

width	ディスプレイの幅(ピクセル)
height	ディスプレイの高さ(ピクセル)
availWidth	タスクバーなどの部分を除いたディスプレイの幅(ピクセル)
availHeight	タスクバーなどの部分を除いたディスプレイの高さ(ピクセル)
pixelDepth	ディスプレイ深度(bits/pixel)
colorDepth	ディスプレイ色深度(nビットカラー)

【メソッド】

なし

【イベントハンドラ】

なし

eventオブジェクト(JavaScript1.2~)

イベントの情報を取り扱うオブジェクト。

タグ内に設定されたイベントハンドラからだけでなく、ウィンドウ上のどこからでもイベントを取得することができます。

【用法】

```
event.property
```

【プロパティ】

(JavaScript1.2)

type	イベントのタイプ
layerX	イベントが発生したレイヤー上のX座標(ピクセル)
layerY	イベントが発生したレイヤー上のY座標(ピクセル)
pageX	イベントが発生したページ上のX座標(ピクセル)
pageY	イベントが発生したページ上のy座標(ピクセル)
screenX	イベントが発生したディスプレイ上のX座標(ピクセル)
screenY	イベントが発生したディスプレイ上のY座標(ピクセル)
which	マウスボタンが押されたり、押されたキーのASCIIの値の数字を表す
modifiers	マウス又はキーイベント発生時のモディファイキー(修飾キー)の種類(ALT_MASK,CONTROL_MASK,SHIFT_MASK,META_MASK)
data	DragDropイベントが発生した時の、ドロップされたオブジェクトのURL

【メソッド】

なし

【イベントハンドラ】

全てのイベントハンドラ及びイベントタイプ

windowオブジェクト

ブラウザ自身や、アラートウィンドウなどの各種ダイアログボックスの情報を提供したり、操作を行うオブジェクト。

documentオブジェクト、frameオブジェクト、historyオブジェクト、locationオブジェクトなど、多くのオブジェクトを含んでいます。

最上層のオブジェクトなので"window"は省略可能ですが、"open()"や"close()"など、他のオブジェクトと同じ名称のコマンドもあるので、混乱を避けるためにも"window"を記述したほうがよいでしょう。

JavaScript1.2 から、ウィンドウ自身のサイズや表示位置を、より細かく設定できるようになりました。

【用法】

```
window.property
window.method()
self.property
self.method()
top.property
top.methodN()
```

```
parent.property
parent.method()
windowVar.property
windowVar.methodName()
property
method()
```

【プロパティ】

(JavaScript1.0)

defaultStatus	デフォルトのステータス行のメッセージ
length	ウィンドウ内のフレーム数
name	ウィンドウの名前
parent	ウィンドウ内のフレームの親フレーム
self	現在のウィンドウ自身windowと同じ
status	ステータス行のメッセージ
top	1番手前にあるウィンドウ
window	現在のウィンドウ自身。selfと同じ
document	documentオブジェクト
Frame(s)	Frameオブジェクト及び配列
history	historyオブジェクト及び配列
location	locationオブジェクト

(JavaScript1.1)

closed	ウィンドウが閉じられている状態
opener	open()で開かれたウィンドウから元のウィンドウを参照する

(JavaScript1.2)

innerHeight	ウィンドウの表示領域の高さ(ピクセル)
innerWidth	ウィンドウの表示領域の幅(ピクセル)
locationbar	ロケーションバー
menubar	ウィンドウのメニューバー
outerHeight	ウィンドウの外周の高さ(ピクセル)
outerWidth	ウィンドウの外周の幅(ピクセル)
pageXOffset	ウィンドウのX座標の位置
pageYOffset	ウィンドウのY座標の位置
personalbar	ウィンドウのパーソナルバー
scrollbars	ウィンドウのスクロールバー
statusbar	ウィンドウのステータスバー
toolbar	ウィンドウのツールバー

【メソッド】

(JavaScript1.0)

alert()	警告用ダイアログボックスを開く
clearTimeout()	setTimeoutメソッドの停止
close()	ウィンドウを閉じる
confirm()	確認ボタン付きダイアログボックスを開く
open()	新しいウィンドウを開く
prompt()	入力欄付きのダイアログボックスを開く
setTimeout()	ミリ秒単位で指定した時間後に実行する
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

blur()	フォーカスを移動する
focus()	フォーカスをあたえる
scroll()	ウィンドウをスクロールする
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

back()	1つ前のURL に進む
disableExternal Capture	外部からのイベントキャプチャを無効にする
enableExternal Capture	異なったロケーション(サーバー)からのイベントキャプチャを許す(Signed Scriptなどと合わせて使用する)
find()	ウィンドウの中に指定した文字列があるか(あればtrueなければfalse)
forward()	1つ先のURL に進む
home()	homeに指定したURL に進む
moveBy()	ウィンドウを移動する(水平方向と垂直方向の移動量をピクセルで指定)
moveTo()	ウィンドウを移動する(ウィンドウの左上の角を基準にピクセルで指定)
resizeBy()	ウィンドウをリサイズする(底辺の角を基準に水平方向、垂直方向をピクセルで指定)
resizeTo()	ウィンドウをリサイズする(ウィンドウのサイズをピクセルで指定)
scrollBy()	ウィンドウをスクロールする(ウィンドウの表示領域の水平方向と垂直方向に対してピクセルで指定)
scrollTo()	ウィンドウをスクロールする(ウィンドウの左上の角を基準にピクセルで指定)
stop()	読み込みを中止する
captureEvents()	すべてのタイプのイベントを取得する
setInterval()	一定時間(1000分の1秒)ごとに指定された処理を繰り返す
clearInterval()	setIntervalメソッドの停止
handleEvent()	イベント取り扱い者を特定する
print()	プリントする
releaseEvents()	別階層のイベントにイベントを渡す
routeEvent()	取得したイベントと同じ階層のイベント

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【open()の属性オプション】

(JavaScript1.0)

* = [=*yes*|*no*]|[=*1*|*0*]

toolbar *	ツールバー
location *	ロケーションボックス

directories *	ディレクトリーボタン
status *	ステータスバー
menubar *	メニューバー
scrollbars *	スクロールバー
resizable *	リサイズボックス
width=pixels	ウィンドウの横幅
height=pixels	ウィンドウの縦幅

(JavaScript1.2)
* = [=*yes*|*no*]|[=*1*|*0*]

alwaysLowered *	他のウィンドウより前に来るウィンドウ
alwaysRaised *	他のウィンドウより後ろに来るウィンドウ
dependent *	現在のウィンドウの子ウィンドウ
hotkeys *	ホットキーを無力にする
innerWidth =pixels	ウィンドウの表示領域の横幅(widthから変更)
innerHeight =pixels	ウィンドウの表示領域の高さ(heightから変更)
outerWidth =pixels	ウィンドウの外周の幅
outerHeight =pixels	ウィンドウの外周の高さ
screenX=pixels	ディスプレイ左上からのX軸の位置
screenY=pixels	ディスプレイ左上からのY軸の位置
titlebar *	タイトルバーの表示
z-lock *	フォーカスが移っても他のウィンドウの前に来ないウィンドウ

【イベントハンドラ】

(JavaScript1.0)
onLoad
onUnload

(JavaScript1.1)
onBlur
onError
onFocus

frameオブジェクト

frameオブジェクトの情報を提供したり、操作を行うオブジェクト。

frame オブジェクトは、それ自体が window オブジェクトのプロパティですが、最上層のオブジェクトである"window"は省略可能です。

ウィンドウ内のフレームの配列を作成します。フレームの情報を取得し、利用することによって、フレーム操作の幅を広げることができます。

【用法】

```
frame名.property
frames[インデックス].property
window.property
self.property
parent.property
```

【プロパティ】

(JavaScript1.0)

frames	フレーム配列
name	フレーム名
length	フレームの数
parent	現在のフレームの親フレーム
self	現在のフレーム自身
window	現在のフレームのウィンドウ自身

(JavaScript1.1)
なし

【メソッド】

(JavaScript1.0)

clearTimeout()	setTimeoutメソッドの停止
setTimeout()	ミリ秒(1000分の1秒)単位で指定された時間後に命令を実行する
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

blur()	フォーカスを移動する
focus()	フォーカスを与える
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

setInterval()	一定時間(1000分の1秒)ごとに指定された処理を繰り返す
clearInterval()	setIntervalメソッドの停止
handleEvent()	イベント取り扱い者を特定する
print()	プリントする

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)
なし

(JavaScript1.1)
onblur
onFocus

documentオブジェクト

HTMLファイル、及びそこに記述されるドキュメントの情報を提供したり、操作を行うオブジェクト。

layerオブジェクト、Linkオブジェクト、Imageオブジェクト、Areaオブジェクト、Anchorオブジェクト、Appletオブジェクト、Formオブジェクトなど、多くのオブジェクトを含んでいます。

documentオブジェクトはそれ自体がwindowオブジェクトのプロパティですが、最上層のオブジェクトである"window"は省略可能です。

【用法】

```
(window.)document.property
(window.)document.method()
```

【プロパティ】

(JavaScript1.0)

alinkColor	アクティブリンクの色
bgColor	背景の色
cookie	クッキーファイルの情報
fgColor	テキストの色
lastModified	ファイルの最終更新日時
linkColor	リンクの色
referrer	リンク元のURL
title	ドキュメントのタイトル
URL	カレントのURL
vlinkColor	すでに行ったことのあるリンクの色
anchor(s)	アンカーオブジェクト及び配列
form(s)	フォームオブジェクト及び配列
link(s)	リンクオブジェクト及び配列

(JavaScript1.1)

domain	カレントのドメイン名(tainting状態時のみ使用可能)
applet(s)	アプレット配列
Area	エリアオブジェクト
embeds	プラグイン配列
image(s)	イメージオブジェクト及び配列

(JavaScript1.2)

layer(s)	レイヤーオブジェクト及び配列

【メソッド】

(JavaScript1.0)

close()	ドキュメントストリームを閉じる
open()	ドキュメントストリームを開く
write()	テキストを書き出す
writeln()	テキストを改行付きで書き出す
eval()	数値に変更する
toString()	オブジェクトを文字列にする

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.2)

getSelection()	選択範囲内に含まれている文字列を返す
captureEvents()	すべてのタイプのイベントを取得する
releaseEvents()	別階層のイベントにイベントを渡す
routeEvent()	取得したイベントと同じ階層のイベント

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

【JavaScript1.1で削除されたプロパティ・メソッド】

location
clear()

historyオブジェクト

ブラウザがロードしたページの来歴を提供するオブジェクト。

historyオブジェクトはそれ自体がwindowオブジェクトのプロパティですが、最上層のオブジェクトである"window"は省略可能です。

注意 削除されたプロパティ・メソッドについて

現在Netscape社の「JavaScript Guide」では、JavaScript1.0にはあった「locationプロパティ」と「clear()メソッド」が姿を消しています。

「locationプロパティ」は、以前より「locationプロパティは将来的には使えなくなるので、URLプロパティを使うように」とのアナウンスがされており、これはおそらくlocationオブジェクトとの混乱を避けるための処置だと思われます。

また「clear()メソッド」に関しては、「open()メソッド」などを利用して似たような効果を出すことが可能なため、削除されたものと思われます。

なお、この2つのコマンドは、今のところNetscape Navigator3.0で使用可能なようですが（Windows版では"clear()"は正常に動かない可能性があります）、今後のことを考えてなるべく使用しないことをお薦めします。

【用法】

```
(window.)history.property
(window.)history.method()
(window.)history[インデックス]
```

【プロパティ】

(JavaScript1.0)

length	来歴の数

(JavaScript1.1)

current	現在の来歴(tainting状態時のみ使用可能)
next	次の来歴(tainting状態時のみ使用可能)
previous	1つ前の来歴(tainting状態時のみ使用可能)

【メソッド】

(JavaScript1.0)

back()	1つ前のページに進む
forward()	1つ先のページへ進む
go ()	指定された分ページを移動する
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

locationオブジェクト

表示されているページのURLに関する情報を提供するオブジェクト。

location オブジェクトはそれ自体が window オブジェクトのプロパティですが、最上層のオブジェクトである"window"は省略可能です。

また、locationオブジェクトは読み出しだけでなく、動的にURLを変更することが可能です。

【用法】

```
(window.)location.property
(window.)location.method()
```

【プロパティ】

(JavaScript1.0)

hash	アンカー
host	URLのホスト名とポート番号部分
hostname	ホストコンピュータ名部分
href	URL
pathname	URLのパス名部分
port	URLのポート番号部分
protocol	URLのプロトコル部分
search	URLの問い合わせ部分

(JavaScript1.1)

なし

【メソッド】

(JavaScript1.0)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

reload()	リロードする
replace()	現在のURLを置き換える
valueOf()	オブジェクトの値を返す

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

Linkオブジェクト

HTMLファイル内のリンクの配列を作成し、リンク1つ1つの情報を提供するオブジェクト。

Link オブジェクトは、それ自体が document オブジェクトのプロパティです。

【用法】

```
document.links[インデックス].property
document.links.length
```

【プロパティ】

(JavaScript1.0)

hash	アンカー名
host	リンク先のURLのホスト名とポート番号部分
length	リンク数
hostname	リンク先のURLのホストコンピュータ名部分
href	リンク先のURL
pathname	リンク先のURLのパス名部分
port	リンク先のURLのポート番号部分
protocol	リンク先のURLのプロトコル部分
search	リンク先のURLの問い合わせ部分
target	リンク先のトランジット属性

(JavaScript1.1)
なし

【メソッド】
(JavaScript1.0)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】
(JavaScript1.0)
onClick
onMouseOver

(JavaScript1.1)
onMouseOut

Anchorオブジェクト(配列)

HTMLファイル内のAnchorの配列を作成するオブジェクト。

Anchorオブジェクトは、それ自体がdocumentオブジェクトのプロパティです。

【用法】

```
document.anchors[インデックス]
document.anchors.length
```

【プロパティ】
(JavaScript1.0)

length	アンカーの数

(JavaScript1.1)
なし

【メソッド】
(JavaScript1.0)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】
なし

Formオブジェクト

Formに関する情報を提供したり、操作を行うオブジェクト。

Formオブジェクトは、それ自体がdocumentオブジェクトのプロパティです。

HTMLの<FORM>タグを、JavaScriptのオブジェクトとして取り扱います。データ入力を受け付けて送信するだけでなく、入力内容のチェックや入力されたデータの計算をブラウザ側で行ったり、リアルタイムでフォームの内容を変化させるなど、フォームの利用方法を広げることが可能です。

Textareaオブジェクト、Textオブジェクト、File Uploadオブジェクト、Passwordオブジェクト、Hiddenオブジェクト、Submitオブジェクト、Resetオブジェクト、Radioオブジェクト、Checkboxオブジェクト、Buttonオブジェクト、Selectオブジェクトなど、多くのオブジェクトを含んでいます。

【用法】

```
form名.property
form名.method()
forms[インデックス].property
forms[インデックス].method()
```

【プロパティ】
(JavaScript1.0)

action	フォーム内のデータの送り先
elements	フォーム内の内容(配列)
encoding	フォームのMINEエンコード
length	フォームエレメントの数
name	オブジェクト名
method	フォームの送信方法
target	ターゲットウィンドウ
Button	Buttonオブジェクト
Checkbox	Checkboxオブジェクト
Hidden	Hiddenオブジェクト
Password	Passwordオブジェクト
Radio	Radioオブジェクト
Reset	Resetオブジェクト
Select	Selectオブジェクト
Submit	Submitオブジェクト
Text	Textオブジェクト
Textarea	Textareaオブジェクト

(JavaScript1.1)

FileUpload	FileUploadオブジェクト

【メソッド】

(JavaScript1.0)

submit()	サブミットボタンが押されたのと同じ
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

reset()	リセットボタンが押された状態と同じ
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)

onSubmit

(JavaScript1.1)

onReset

Buttonオブジェクト

フォームのButtonに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<INPUT TYPE="button">タグを使って作成されたボタン型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
button名.property
button名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

name	オブジェクト名
value	オブジェクト内の値

(JavaScript1.1)

type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)

click()	クリックと同じ
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

blur()	フォーカスを移動する
focus()	フォーカスをあたえる
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)

onClick

(JavaScript1.1)

onBlur
onFocus

Checkboxオブジェクト

フォームのCheckboxに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<INPUT TYPE="Checkbox">タグを使って作成されたチェックボックス型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
checkbox名.property
checkbox名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

checked	チェックされている状態
defaultChecked	デフォルトでチェックされている状態
name	オブジェクト名
value	オブジェクト内の値

(JavaScript1.1)

type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)

click()	クリックと同じ
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

blur()	フォーカスを移動する
focus()	フォーカスをあたえる
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)
onClick

(JavaScript1.1)
onBlur
onFocus

FileUploadオブジェクト(JavaScript1.1～)

フォームのFileUploadに関する情報を提供したり、操作を行うオブジェクト。

HTML の <INPUT TYPE="FileUpload"> タグを使って作成されたファイル選択欄型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
FileUpload名.property
FileUpload名.method()
```

【プロパティ】

(JavaScript1.0)
未対応

(JavaScript1.1)

name	オブジェクト名
value	オブジェクト内の値
type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)
未対応

(JavaScript1.1)

blur()	フォーカスを移動する
focus()	フォーカスをあたえる
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)
未対応

(JavaScript1.1)
onBlur
onChange
onFocus

Hiddenオブジェクト

フォームのHiddenに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<INPUT TYPE="Hidden">タグを使って作成された、隠しテキストボックス型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
Hidden名.property
form名.elements[インデックス].property
```

【プロパティ】

(JavaScript1.0)

name	オブジェクト名
value	オブジェクト内の値

(JavaScript1.1)

type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

Passwordオブジェクト

フォームのPasswordに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<INPUT TYPE="Password">タグを使って作成されたパスワード入力用テキストボックス型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
Password名.property
Password名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

defaultValue	デフォルトのオブジェクト内の値
name	オブジェクト名
value	オブジェクト内の値

(JavaScript1.1)

type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)

select()	選択された状態
blur()	フォーカスを移動する
focus()	フォーカスをあたえる
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)
なし

(JavaScript1.1)
onBlur
onFocus

Radioオブジェクト

フォームのRadioに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<INPUT TYPE="Radio">タグを使って作成されたラジオボタン型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
Radio名[インデックス].property
Radio名[インデックス].method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

checked	選択されている状態
defaultChecked	デフォルトで選択されている状態
length	オブジェクトの数
name	オブジェクト名
value	オブジェクト内の値

(JavaScript1.1)

type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)

click()	クリックと同じ
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

blur()	フォーカスを移動する
focus()	フォーカスを与える
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)
onClick

(JavaScript1.1)
onBlur
onFocus

Resetオブジェクト

フォームのResetに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<INPUT TYPE="Reset">タグを使って作成されたリセットボタン型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
Radio名[インデックス].property
Radio名[インデックス].method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

name	オブジェクト名
value	オブジェクト内の値

(JavaScript1.1)

type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)

click()	クリックと同じ
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

blur()	フォーカスを移動する
focus()	フォーカスをあたえる
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)

onClick

(JavaScript1.1)

onBlur
onFocus

Selectオブジェクト

フォームのSelectに関する情報を提供したり、操作を行うオブジェクト。

HTML の <SELECT> タグと <OPTION> タグを使って作成された、選択欄型のフォームとその中の選択項目を、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
select名[インデックス].property
select名[インデックス].method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

<オプション>

```
select名.options[インデックス].property
form名.elements[インデックス].options[インデックス].property
option名.property
```

<オプション配列>

```
select名.options
select名.options[インデックス]
select名.options.length
```

【プロパティ】

(JavaScript1.0)

length	オプションの数
name	オブジェクト名
options	オプション項目
selectedIndex	選択されている項目

(JavaScript1.1)

type	オブジェクトのタイプ

【オプション(配列)】

(JavaScript1.0)

defaultSelected	デフォルトで選択されている項目
index	項目の位置
selected	選択されている状態
value	オブジェクト内の値

(JavaScript1.1)

prototype	新しいプロパティの作成

【メソッド】

(JavaScript1.0)

click()	クリックと同じ
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

blur()	フォーカスを移動する
focus()	フォーカスをあたえる
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)

onChange

(JavaScript1.1)

onBlur

onFocus

Submitオブジェクト

フォームのSubmitに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<INPUT TYPE="Submit">タグを使って作成されたフォーム内容送信用ボタン型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
Submit名.property
Submit名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

name	オブジェクト名
value	オブジェクト内の値

(JavaScript1.1)

type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)

click()	クリックと同じ
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

blur()	フォーカスを移動する
focus()	フォーカスをあたえる
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)

onClick

(JavaScript1.1)

onBlur

onFocus

Textareaオブジェクト

フォームのTextareaに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<TEXTAREA>タグを使って作成されたテキストエリア型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
textarea名.property
textarea名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

defaultValue	デフォルトのオブジェクトの値
name	オブジェクト名
value	オブジェクト内の値

(JavaScript1.1)

type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)

blur()	フォーカスを移動する
focus()	フォーカスをあたえる
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)

onBlur
onChange
onFocus
onSelect

(JavaScript1.1)

なし

≫Textオブジェクト

フォームのTextに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<INPUT TYPE="Text">タグを使って作成されたテキスト入力欄型のフォームを、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
Text名.property
Text名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

defaultValue	デフォルトのオブジェクトの値
name	オブジェクト名
value	オブジェクト内の値

(JavaScript1.1)

type	オブジェクトのタイプ

【メソッド】

(JavaScript1.0)

blur()	フォーカスを移動する
focus()	フォーカスをあたえる
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)

onBlur
onChange
onFocus
onSelect

(JavaScript1.1)

なし

Areaオブジェクト(JavaScript1.1)

イメージマップの定義や、その情報を提供するオブジェクト。

Areaオブジェクトは、Linkの配列内にあります。したがって、リンクを参照する場合は、「document.links[インデックス]」を使用します。

【用法】

```
area名.property
document.links[インデックス].property
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

hash	アンカー名
host	リンク先のURLのホスト名とポート番号部分
hostname	リンク先のURLのホストコンピュータ名部分
href	リンク先のURL
pathname	リンク先のURLのパス名部分
port	リンク先のURLのポート番号部分
protocol	リンク先のURLのプロトコル部分
search	リンク先のURLの問い合わせ部分
target	リンク先のトランジット属性

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.0)

未対応

(JavaScript1.1)

onMouseOver
onMouseOut

Imageオブジェクト(JavaScript1.1~)

画像に関する情報を提供したり、操作を行うオブジェクト。

画像ファイルの情報を提供する以外に、画像を後から置き換えることも可能です。

これにより、画像をアニメーションさせたり、イベントハンドラと組み合わせることによって、インタラクティブに画像を取り扱うことができます。

ビルトインオブジェクトのように、newを使ってオブジェクトを新たに作成することができます。

【オブジェクトの作成】

```
image名 = new Image(画像の幅, 画像の高さ)
```

【用法】

```
オブジェクト名.property
document.images[インデックス].property
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

border	borderで設定された値
complete	画像の読み込みが終了しているかどうか
height	イメージの高さ
hspace	heightで設定された値
length	イメージの数
lowsrc	lowsrcで設定している画像ファイルのURL
name	イメージの名前
prototype	新しいプロパティの作成
src	画像ファイルのURL
vspace	vspaceで設定された値
width	画像の幅

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

handleEvent()	イベント取り扱い者を特定する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

(JavaScript1.1)

onAbort
onError
onLoad

layerオブジェクト(JavaScript1.2~)

layerに関する情報を提供したり、操作を行うオブジェクト。

Netscape Navigator4.0で新たに追加された、<LAYER>タグに関連したJavaScriptです。

レイヤーのshow(見える状態)とhide(隠された状態)の動的な切り替えや、指定位置への移動など、レイヤーによりインタラクティブな要素を付け加えることができます。

layerオブジェクトは、documentオブジェクトのプロパティです。

【用法】

```
layer名.property
layer名.method(数値)
document.layer名
document.layers[インデックス]
document.layers["layer名"]
(ネストされているレイヤーのプロパティを使う時)
document.layers["親layer"].layers
["子layer"]
```

【プロパティ】

(JavaScript1.2)

name	オブジェクト名の設定
width	レイヤーのコンテンツの横幅(ピクセル)
height	レイヤーのコンテンツの高さ(ピクセル)
left	画面左からのレイヤーの位置(ピクセル)

top	画面上からのレイヤーの位置(ピクセル)
visibility	レイヤーがshow(見える状態)かhide(隠された状態)可かを返す
zIndex	レイヤーのZ-Indexの値を返す
pageX	ページ上のX座標(ピクセル)
pageY	ページ上のy座標(ピクセル)
x	"Layer.left. "と同義
y	"Layer.top."と同義
clip.top	レイヤー内のクリップの上からの位置
clip.left	レイヤー内のクリップの左からの位置
clip.right	レイヤー内のクリップの右からの位置
clip.bottom	レイヤー内のクリップの下からの位置
clip.width	レイヤー内のクリップの横幅
clip.height	レイヤー内のクリップの高さ
background	レイヤーのバックグラウンドイメージの指定
bgColor	レイヤーのバックグラウンドカラーの指定
siblingAbove	兄弟レイヤー内で1つ上のレイヤーを参照する、もし1番先頭だったら"null"を返す
siblingBelow	兄弟レイヤー内で1つ下のレイヤーを参照する、もし1番後ろだったら"null"を返す
above	1つ上のレイヤーを参照する、もし1番先頭だったら"null"を返す
below	1つ下のレイヤーを参照する、もし1番後ろだったら"null"を返す
parentLayer	符合するレイヤー状況、親のレイヤーだったら"null"を返す
layers	LAYER配列
src	レイヤーのソースを外部から呼び出す時のURL

【メソッド】

(JavaScript1.2)

moveBy()	現在の表示位置からのレイヤーの表示位置移動(ピクセル)
moveTo()	画面の左上からのレイヤーの表示位置移動(ピクセル)
moveTo Absolute()	レイヤーコンテンツ内でのレイヤー位置移動(ピクセル)
resizeBy()	レイヤーのサイズを指定された分だけ変更する(ピクセル)
resizeTo()	レイヤーのサイズを指定されたサイズへ変更する(ピクセル)
moveAbove()	レイヤーを前に出す
moveBelow()	レイヤーを後ろに入れる
load()	外部からレイヤーのHTMLファイルを読み込む
captureEvents	すべてのタイプのイベントを取得する
handleEvent()	イベント取り扱い者を特定する
releaseEvents()	別階層のイベントにイベントを渡す
routeEvent()	取得したイベントと同じ階層のイベント

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

onMouseOver
onMouseOut
onLoad
onFocus
onBlur

Appletオブジェクト(配列)(JavaScript1.1～)

Applet関連のオブジェクト及び配列。

【用法】

```
document.applets[インデックス]
document.applets.length
```

【プロパティ】

(JavaScript1.1)

length	アプレットの数(配列数)

【メソッド】

(JavaScript1.1)

JavaAppletの全ての"public method"

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える
valueOf()	オブジェクトの値を返す

embeds配列(JavaScript1.1～)

プラグイン関連のオブジェクト及び配列。

【用法】

```
document.embeds[インデックス]
document.embeds.length
```

【プロパティ】

(JavaScript1.1)

length	プラグインの数(配列数)

【メソッド】

(JavaScript1.1)

各プラグイン独自のメソッド

7. ビルトインオブジェクト

Dateオブジェクト

日付や時間の取得や設定を行うオブジェクト。

ブラウザの起動時に、そのマシンから時間の情報を取得し、加工します。各マシンが置かれている場所のローカルタイムを扱うので、CGIでは難しい世界各地のローカルタイムに合わせた処理を行うことができます。

【オブジェクトの作成】

```
オブジェクト名 = new Date()
オブジェクト名 = new Date("month day,
year hours:minutes:seconds")
オブジェクト名 = new Date(year, month
, day)
オブジェクト名 = new Date(year, month
, day, hours, minutes, seconds)
```

(JavaScript1.3)

```
オブジェクト名 = new Date (year,month,
day, [,hours[,minutes[,seconds[,
milliseconds]]]])
オブジェクト名 = new Date.UTC(year,
month, date, [,hours[,minutes[,
seconds[, milliseconds]]]])
```

【用法】

```
オブジェクト名.property
オブジェクト名.method()
```

【プロパティ】

(JavaScript1.0)

なし

(JavaScript1.1)

constructor	オブジェクトの作成元
prototype	オブジェクトのプロトタイプを作成

【メソッド】

(JavaScript1.0)

getDate()	日を返す(1～31)
getDay()	曜日を返す(0～6)
getHours()	時間を返す(0～23)
getMinutes()	分を返す(0～59)
getMonth()	月を返す(0～11)
getSeconds()	秒を返す(0～59)
getTime()	1970年1月1日0時0分0秒からの経過時間をミリ秒単位で返す
getTimezoneOffset()	グリニッジ標準時(GTM)とローカル時間の差を分単位で返す
getYear()	年を返す
parse()	指定された日付と時間と1970年1月1日0時0分0秒との間の時間をミリ単位で返す
setDate()	日付の設定をする
setHours()	時間の設定をする
setMinutes()	分の設定をする
setMonth()	月の設定をする
setSeconds()	秒の設定をする
setTime()	ミリ秒単位で日付けと時間の設定をする
setYear()	年の設定をする
toGMTString()	日付と時間をグリニッジ標準時(GTM)の文字列に変換する
toLocaleString()	日付と時間をローカルタイムの文字列で返す
UTC()	1970年1月1日0時0分0秒からの経過時間をグリニッジ標準時(GTM)を基にミリ秒単位で返す
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.3)

getFullYear()	4桁の年を返す
setFullYear()	4桁の年を設定する
getMilliseconds()	ミリセコンドを返す(0～999)
setMilliseconds()	ミリセコンドを設定する
getUTCFullYear()	4桁のUTC時間の年を返す
getUTCMonth()	UTC時間の月を返す(0～11)
getUTCDate()	UTC時間の日を返す(1～31)
getUTCDay()	UTC時間の曜日を返す(0～6)
getUTCHours()	UTC時間の時間を返す(0～23)
getUTCMinutes()	UTC時間の分を返す(0～59)
getUTCSeconds()	UTC時間の秒を返す(0～59)
getUTCMilliseconds()	UTC時間のミリセコンドを返す(0～999)
toUTCString()	UTC時間を文字列で返す

setUTCFull Year()	4桁のUTC時間の年を設定する
setUTCMonth()	4桁のUTC時間の月を設定する
setUTCDate()	4桁のUTC時間の日を設定する
setUTCHours()	UTC時間の時間を設定する
setUTC Minutes()	UTC時間の分を設定する
setUTC Seconds()	UTC時間の秒を設定する
setUTCMilli seconds()	UTC時間のミリセコンドを設定する
toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

Mathオブジェクト

数値計算関連のオブジェクト。"Math"の後に利用するプロパティやメソッドを記述して使用します。

それらを組み合わせて使うことによって、サーバーにデータを渡すことなく、いろいろな計算をブラウザで行うことができます。

【用法】

```
Math.property
Math.method()
```

【プロパティ】

(JavaScript1.0)

E	自然対数の底
LN2	eを底とする2の自然対数
LN10	eを底とする10の自然対数
LOG2E	2を底とする自然対数
LOG10E	10を底とする自然対数
PI	円周率
SQRT1_2	1/2の平方根
SQRT2	2の平方根

(JavaScript1.1)

なし

【メソッド】

(JavaScript1.0)

abs()	0からの絶対値を返す
acos()	アークコサインを返す
asin()	アークサインを返す
atan()	アークタンジェントを返す
atan2()	座標から角度を返す
ceil()	最も近くて大きい整数を返す
cos()	コサインを返す
exp()	対数を返す
floor()	最も近くて小さい整数を返す
log()	自然対数を返す
max()	比較して大きい方の数値を返す
min()	比較して小さい方の数値を返す
pow()	乗算をする
random()	乱数を発生する(Netscape Navigator 2.xはUNIX版のみ)
round()	四捨五入する
sin()	サインを返す
sqrt()	平方根を返す
tan()	タンジェントを返す
eval()	数値に変更する
toString()	オブジェクトを文字列にする

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

stringオブジェクト

文字列に対して修飾や検索などの操作を行います。文字列の後にプロパティやメソッドを付けて使用します。

JavaScript1.1からnewを使って新しくオブジェクトを作成することができるようになりました。

【新たにオブジェクトを作る(JavaScript1.1)】

```
オブジェクト名 = new String(文字列)
```

【用法】

```
文字列.property
文字列.method()
```

【プロパティ】

(JavaScript1.0)

length	文字の数

(JavaScript1.1)

constructor	オブジェクトの作成元
prototype	オブジェクトのプロトタイプを作成

【メソッド】

(JavaScript1.0)

anchor()	アンカーを設定する
big()	文字を大きくする
blink()	文字を点滅させる
bold()	太字にする
charAt()	文字を抜き出す
fixed()	等幅文字にする
fontcolor()	文字色を指定する
fontsize()	フォントサイズを指定する
indexOf()	文字を検索する
italics()	斜体文字にする
lastIndexOf()	文字列の後ろから検索する
link()	リンクを作成する
small()	文字を小さくする
strike()	削除文字にする
sub()	下付文字にする
substring()	文字列の途中を抜き出す
sup()	上付文字にする
toLowerCase()	大文字を小文字にする
toUpperCase()	小文字を大文字にする
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える

(JavaScript1.1)

split()	文字列を分割する
valueOf()	オブジェクトの値を返す

(JavaScript1.2)

charCodeAt()	ISO-Latin-1 のコード番号を返す
fromCharCode()	ISO-Latin-1のコード番号を文字に直す
substr()	文字を抜き出す(n番からm個の文字)
match()	指定したパターンと同じパターンをを見つける
replace()	指定したパターンを置き換える
split()	指定したパターンの部分で文字列を分ける
match()	指定したパターンと同じパターンを見つける
replace()	指定したパターンを置き換える
split()	指定したパターンの部分で文字列を分ける

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

Arrayオブジェクト (JavaScript1.1〜)

配列を作成し、操作するオブジェクト。複雑な手順なしに配列が作成できます。

JavaScript1.1 から追加されたオブジェクトです。一部は Netscape Navigator2.0 から利用可能ですが、正式なサポートではありません。

【オブジェクトの作成】

```
arrayオブジェクト名 = new Array(配列の数)
arrayオブジェクト名 = new Array(0番目の要素,1番目の要素, ...,n番目の要素)
```

【用法】

```
オブジェクト名.property
オブジェクト名.method()
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

length	配列の数
constructor	オブジェクトの作成元
prototype	配列のプロトタイプ

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

join()	要素を文字列にする
reverse()	要素の順番を逆にする
sort()	要素をソートする
eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える
valueOf()	オブジェクトの値を返す

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

functionオブジェクト(JavaScript1.1～)

関数の作成や操作を行うオブジェクト。

JavaScript1.1からnewを使って新しい関数のオブジェクトを作成できるようになりました。

argumentsはJavaScript1.0から存在しましたが、JavaScript1.1から正式にfunctionオブジェクトのプロパティになりました。

【オブジェクトの作成】

```
オブジェクト名= new Function ([要素1,
要素2,...要素n], functionの働き)
```

【用法】

```
オブジェクト名.property
オブジェクト名.method(数値)
```

【プロパティ】

(JavaScript1.0)
未対応

(JavaScript1.1)

caller	ファンクションがどこから呼ばれたか
arguments	ファンクションの要素[配列]
arguments.caller	ファンクションがどこから呼ばれたか
arguments.length	ファンクション配列の要素数
length	用意されているファンクションの要素数
constructor	オブジェクトの作成元
prototype	ファンクションのプロトタイプ

(JavaScript1.2)

arguments.callee	ファンクションの内容
arity	用意されているファンクションの要素数

【メソッド】

(JavaScript1.0)
未対応

(JavaScript1.1)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える
valueOf()	オブジェクトの値を返す

(JavaScript1.3)

call()	異なるオブジェクトをコールする
apply()	異なるオブジェクトをコールする(値を引き渡すことが可能)
toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

Objectオブジェクト

ユーザー独自の新しいオブジェクトを作成するオブジェクト。

【オブジェクトの作成】

```
オブジェクト名 = new Object()
```

【用法】

```
オブジェクト名.property
オブジェクト名.method()
```

【プロパティ】

(JavaScript1.0)
なし

(JavaScript1.1)

constructor	オブジェクトの作成元
prototype	オブジェクトのプロトタイプを作成

【メソッド】

(JavaScript1.0)

toString()	オブジェクトを文字列に変える

(JavaScript1.1)

valueOf()	オブジェクトの値を返す

(JavaScript1.2)

unwatch()	プロパティの監視を解除する
watch()	プロパティを監視する

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

Booleanオブジェクト (JavaScript1.1～)

真（true）の値と、偽（false）の値を作成するオブジェクト。

【オブジェクトの作成】

```
オブジェクト名 = new Boolean()
```

【用法】

```
オブジェクト名.property
オブジェクト名.method
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

constructor	オブジェクトの作成元
properties	新しいプロパティの作成

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える
valueOf()	オブジェクトの値を返す

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

Numberオブジェクト (JavaScript1.1～)

最大値などの属性を持った数値を作成するオブジェクト。

【オブジェクトの作成】

```
オブジェクト名 = new Number()
```

【用法】

```
オブジェクト名.property
オブジェクト名.method
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

MAX_VALUE	最大値を表す数値
MIN_VALUE	最小値を表す数値
NaN	値数が値では無いことを表す
NEGATIVE_ INFINITY	マイナス方向の限界の値、越えると"overflow"を返す
POSITIVE_ INFINITY	プラス方向の限界の値、越えると"overflow"を返す
constructor	オブジェクトの作成元
properties	新しいプロパティの作成

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える
valueOf()	オブジェクトの値を返す

(JavaScript1.3)

toSource()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

RegularExpressionオブジェクト (JavaScript1.2～)

正規表現関連のオブジェクト。

【オブジェクトの作成】

```
正規表現 = /パターンの設定/[i|g|gi]
regexp = new RegExp("パターンの設定",
["i"|"g"|"gi"])
```

【オプション(省略可能)】

i	大文字、小文字無視
g	完全一致
gi	大文字、小文字無視して完全一致

【プロパティ】

(JavaScript1.2)

global	完全一致
ignoreCase	大文字、小文字無視
lastIndex	次のパターンマッチを始める位置
source	マッチさせるパターン
input ($_でも可)	検索文字列の設定、変更
multiline ($*でも可)	改行コードを無視するかどうか。true(無視しない)、false(無視する)で設定
lastMatch ($&でも可)	パターンがマッチした最後の文字
lastParen ($+でも可)	パターンがマッチした最後のサブストリング
leftContext ($¥Qでも可)	パターンがマッチしたものの直前の文字
rightContext ($'でも可)	パターンがマッチしたものの次の文字
$1~$9	パターンがマッチしたものの一部を蓄える(9個まで)

【メソッド】

(JavaScript1.2)

compile()	ループ文内で使うパターンを作成する
exec()	指定したパターンと同じパターンを見つける
test()	同じパターンがあるかどうかテストする

(JavaScript1.3)

toSource ()	オブジェクトの値を文字列で返す

【イベントハンドラ】

なし

8. Top-Levelプロパティ

オブジェクトに関係なく使用できるプロパティ。

(JavaScript1.3)

Infinity	無限大を表している数値
NaN	数値でない値(Not A Number)
undefined	不確定な値

9. ビルトイン関数(top-level関数)

JavaScriptには、始めから定義されている関数があり、それをビルトイン関数といいます。
ビルトイン関数はオブジェクトに依存することなく、スクリプト内のどこからでも使用することができます。

(JavaScript1.0)

parseInt()	文字列を整数に変更する
parseFloat()	文字列を浮動小数点に変更する
escape()	文字をASCII形式に変換する (unescape()の逆)
unescape()	ASCII形式を文字に変換する (escape()の逆)
isNaN()	値が数値かどうか判断する

(JavaScript1.1)

taint()	データ参照を許可しないようにする
untaint()	taint()を無効にする

※"taint()"及び"untaint()"は、通常のブラウザでは機能しません。また、JavaScript 1.2では削除されました。

(JavaScript1.2)

Number()	オブジェクトを数に変換する
String()	オブジェクトを文字列に変換する

(JavaScript1.3)

isFinite()	有限数かどうか判断する

JavaScript
付録

Netscape Navigatorでスタイルシートの情報を取得する方法

Netscape Navigatorでは、スタイルシートもレイヤーと同じ用法で取り扱うことができます。この時、スタイルシートで設定する「ID」がそのままJavaScriptのオブジェクト名になります。

次のサンプルは、レイヤーオブジェクトの「レイヤーの情報を取得する」(P.416)のレイヤー部分を、スタイルシートに変更したものです。レイヤーと同じように、スタイルシートの情報が取得できることが分かるでしょう。

しかしながら、Netscape Navigatorのスタイルシートのサポートには、多くの問題があります。詳しくは「クロスブラウザDHTMLを作るには」(P.578)で触れているので、そちらも合わせて参考にしてください。

*スタイルシートの情報を取得する(Netscape)

オブジェクト名:STY1
画面左からの位置:200
画面上からの位置:50
画面左からの位置:200
画面上からの位置:50
Z-Indexの値:1
レイヤーの可視属性:show

```
<HTML><HEAD><TITLE></TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
*スタイルシートの情報を取得する(Netscape)<P>
<DIV ID="STY1" STYLE="position:absolute; left:200px; top:50px; width:100px;
height:100px;Z-INDEX:1;visibility:visible;">
   <IMG SRC="image.gif" ALT="image.gif" WIDTH=100 HEIGHT=100>
</DIV>
<SCRIPT LANGUAGE="JavaScript1.2">
<!--
document.write("オブジェクト名：",document.layers["STY1"].name);
document.write("<BR>");
document.write("画面左からの位置：",document.layers.STY1.left);
document.write("<BR>");
document.write("画面上からの位置：",document.layers.STY1.top);
document.write("<BR>");
document.write("画面左からの位置：",document.layers[0].pageX);
document.write("<BR>");
document.write("画面上からの位置：",document.layers["STY1"].pageY);
document.write("<BR>");
document.write("Z-Indexの値：",document.layers["STY1"].zIndex);
document.write("<BR>");
document.write("レイヤーの可視属性：",document.layers["STY1"].visibility);
//-->
</SCRIPT>
</BODY></HTML>
```

DynamicHTMLとは?

DynamicHTMLとは、大雑把にいえば、ページ上にコンテンツの正確な位置や重なりを指定し、そのコンテンツをオブジェクトとして捕らえ(DOM=Document Object Model)、そのオブジェクトをスクリプトなどを使用して、位置や重なりを変化させる技術となります。

本書の場合、「Layerオブジェクト」の「レイヤーの表示・非表示を切り替える」(P.424)や、「レイヤーを移動する」(P.425)、「レイヤーの背景色を変える」(P.427)などが、DynamicHTMLになります。

DynamicHTMLは、Netscape NavigatorでもInternet Explorerでも、バージョン4.0からサポートされました。

しかし、Netscape NavigatorとInternet Explorerでは、DynamicHTMLの実装に微妙な違いがあります。DynamicHTMLを、Netscape Navigatorではコンテンツの位置決めにレイヤーとCSS-Pを、そしてオブジェクトの動きをJavaScriptで設定することで実現しています。Internet Explorer4.0ではコンテンツの位置決めにCSS-Pを、オブジェクトの動きをVBScriptやJavaScriptなどで設定することで実現しています。この実装の違いにより、Netscape NavigatorとInternet Explorer両方で同じように動くDynamicHTMLを作るのは、簡単とはいえず、ちょっとしたコツのようなものが必要です。

また、現在Netscape NavigatorやInternet Explorerで採用されているDOMの用法は、まだW3Cによる標準化が確定する前だったので、それぞれが標準とは違う独自の仕様になっています。

Netscape社の「Mozilla.org」が開発しているレンダリングエンジン「Gecko」は、W3Cの仕様に完全準拠する予定です。したがって、「Gecko」を採用する予定のNetscape Navigator5.0では、DynamicHTMLの設定方法が変わることになります。標準仕様の採用により設定方法が変わることに関しては、「新しい方法を新たに勉強しなければいけない」、「ページを作りなおさなければいけない」など、確かに面倒なことではありますが、みんなで決めた約束ごとの上に運用されているインターネットの世界では、歓迎すべきことでしょう。

また、Netscape Navigator5.0以外にも、「Gecko」の採用を予定しているアプリケーションが多数発表されており、それらのアプリケーションでは、同じDynamicHTMLのソースが、同じように動くことが期待できます。DynamicHTMLを使ったウェブページを作成する場合は、今のうちから「Gecko」に関する情報にも目を向けておくことをお勧めします。

- 「Mozilla.org」
  http://www.Mozilla.org/

- 「The World Wide Web Consortium」
  http://www.w3.org/

クロスブラウザDHTMLを作るには

Netscape NavigatorとInternet ExplorerのDynamicHTMLには全く互換がない、というようなことをよく耳にします。しかし、両者のDHTMLの基本的な部分は同じ所から出ており、ちょっとした工夫で、両方のブラウザで実行可能なDHTMLを作ることが可能です。

DynamicHTMLとは、大雑把にいえば、ページ上にコンテンツの正確な位置や重なりを指定し、そのコンテンツをオブジェクトとして捕らえ(DOM=Document Object Model)、そのオブジェクトをスクリプトなどを使用して、位置や重なりを変化させる技術となります。

DynamicHTMLは、Netscape Navigator4.xではコンテンツの位置決めにレイヤーとCSS-Pを、そしてオブジェクトの動きをJavaScriptで設定することで実現しています。Internet Explorer4.xではコンテンツの位置ぎめにCSS-Pを、オブジェクトの動きをVBScriptとJavaScriptで設定することで実現しています。

つまり、Netscape Navigator4.xとInternet Explorerの4.xのDynamicHTMLでは、CSS-PとJavaScriptの部分が共通で使えることになります。

ただし、それぞれのオブジェクトの指定の仕方がブラウザによって、Netscape Navigatorでは「document.layers["オブジェクト名"]」、Internet Explorerでは「document.all["オブジェクト名"]」と微妙に違います。

そこで、クロスブラウザDHTMLを作成する時は、ブラウザを判断し、それぞれのブラウザに合わせたオブジェクトを使用したスクリプトを実行する必要があります。

ブラウザの判断はJavaScriptを使用すれば可能です。次の例は、navigatorオブジェクトの「appName」プロパティでブラウザ名を取得し、stringオブジェクトの「charAt()」メソッドを使用して1番始めの文字を取り出し、その文字が"M"だった場合はInternet Explorerと判断し、"N"だった場合はNetscape Navigatorと判断しています。

```
if( navigator.appName.charAt(0)=="M" ){ Internet Explorer用のJava
Script }
if( navigator.appName.charAt(0)=="N" ){ Netscape用のJavaScript }
```

実際には、「Internet Explorer用のJavaScript」の部分にInternet Explorer用のスクリプトを、「Netscape Navigator用のJavaScript」の部分にNetscape Navigator用のスクリプトを記述します。

しかし、CSS-Pを使用してページをレイアウトする時、Netscape Navigatorではスタイルシートの表示に問題があり、背景色や背景画像の表示がうまく行なえません。

例えば、次のスタイルシートをInternet ExplorerとNetscape Navigatorで実行した場合、

```
<DIV ID="STY1" STYLE="position:absolute; top:50px; left:50px; back
ground:Green">
position:absolute;left: 200px; top:50px; background:Green
</DIV>
```

Internet Explorer

position:absolute;left: 200px; top:50px; background:Green

Netscape Navigator

position:absolute;left: 200px; top:50px; background:Green

文字を含めた1行分のスタイルシートの背景色が変りますが、Netscape Navigatorでは文字の部分の背景色しか変りません。

この問題は、次のサンプルを実行するともっとはっきりします。

```
<DIV ID="STY1" STYLE="position:absolute; top:50px; left:50px;
 background:Green">
<P>
position:absolute;left: 200px; top:50px; background:Green
</P>
<P>
position:absolute;left: 200px; top:50px; background:Green
</P>
</DIV>
```

Internet Explorer

position:absolute;left: 200px; top:50px; background:Green

position:absolute;left: 200px; top:50px; background:Green

Netscape Navigator

position:absolute;left: 200px; top:50px; background:Green

position:absolute;left: 200px; top:50px; background:Green

このサンプルをNetscape Navigatorで見た場合、文字の後ろしか背景色が変わらず、まだらになってしまうことが解ると思います。

更にこの問題は、次のサンプルのように「width:」プロパティと「height:」プロパティを使用して、スタイルシートのサイズを明示的に設定しても発生します。

```
<DIV ID="STY1" STYLE="position:absolute; top:50px; left:50px; width:400px;
height:200px; background:Green">
<P>
position:absolute; top:50px; left:50px;
</P>
<P>
width:400px; height:200px; background:Green
</P>
</DIV>
```

Internet Explorer

position:absolute; top:50px; left:50px;

width:400px; height:200px; background:Green

Netscape Navigator

position:absolute; top:50px; left:50px;

width:400px; height:200px; background:Green

Internet Explorerでは、設定通りに幅100ピクセル・高さ200ピクセルのスタイルシート全体の背景色が表示されますが、Netscape Navigatorで背景色が変るのは、やはり文字がある部分のみです。

この問題を回避するためは、スタイルシートと同じ大きさの画像をスタイルシートに設定する、ブラウザを判断してInternet Explorerの場合スタイルシートで、Netscape Navigatorの場合レイヤーでページをレイアウトする、などの方法があります。

クロスブラウザDHTMLに関する情報は、Netscape社の「Dynamic HTML Sample Code」の「CROSS-BROWSER」でも公開されています。

- 「Dynamic HTML Sample Code」
  http://developer.netscape.com/docs/examples/dynhtml.html
- 「CROSS-BROWSER」
  http://developer.netscape.com/docs/examples/dynhtml.html#Cross

JavaScriptのありがちなミス

JavaScriptのソース上に問題があった場合、問題の種類と問題があった場所を示したエラー表示のウインドウが開きます。

しかしながら、このウインドウで指摘されている問題があった場所は、実際にはこの場所と別の場所に不具合があったために、エラーが引き起こされた場合もありますので気を付けてください。

エラー表示ウインドウ

Netscape: 警告

JavaScript Error: file:/Macintosh%20HD/Desktop%20Folder/01Num.html, **line 6:**

Nuber is not defined.

OK

また、自分の経験上でスクリプトの作成時に起こしやすいミスをまとめてみました。ブラウザでチェックする前に、以下の項目をもう1度確認してみてください。

記述上のミス

・名称のスペルミス

関数名や変数名は、ユーザーが規定の範囲内で自由に定義できますが、スクリプトを書いている途中で、スペルを間違えたり抜かしてしまうことがよくあります。また、大文字・小文字も区別するので、その点にも注意が必要です。

・オブジェクトの階層のミス

ナビゲータオブジェクトには、オブジェクト自体に階層関係があり、それらのオブジェクトを使用する時は、その階層関係を間違えずに記述して使用する必要があります。特にTextオブジェクトなど、階層の下の方にあるオブジェクトでは、途中のオブジェクトの記述忘れや、順番に気を付けて下さい。

・関数名.変数名の取り違え

長いスクリプトを書いている時、特に変数名などは、値の代入を繰り返していくうちに、どの変数にどの値が代入されたかが、こんがらがってしまうことがあります。これらの名称は、自分自身が見て判別しやすい名称を設定することをお勧めします、また複雑な処理をしているところでは、どういう処理をしているかをコメントに書いておくのもよいでしょう。

・関数名の重複

1つのHTMLファイル内に同名の複数の関数がある場合、1番最後に記述された関数のみが動作し、他の関数は無効になってしまいます。

・"{}"などのカッコや"の閉じ忘れ

処理文が長くなったり、何重にもネスト(入れ子)させていると、処理文を閉じる「}」を付け忘れたり、間違った所に付けてしまうことがよくあります。各階層ごとに段落を下げたり、コメントを付けるなどの工夫することをお勧めします。また、文字列を閉じる「"」なども、文字列が長くなったり、変数と一緒に使ったりしていると、うっかり付け忘れてしまいがちです。

・"と'のネスト(入れ子)の間違い

「" "」で括られた文字列内では、「"」を使わずに「'」を使用します。うっかり「"」を使ってしまうと、JavaScriptは、そこで文字列が終わったと判断して、エラーになってしまいます。特にイベントハンドラ内で、文字列などを設定する時に間違えてしまいがちですので、注意が必要です。

・「=」ではなく「==」

JavaScriptでは、左右の値を評価して値が同じであることを表わす時は、「値A = 値B」ではなく「値A == 値B」と記述します。

文字コード上の問題

・「表示」・「予定」などの文字化けする文字によるエラー

「document.write()」で文字を書き出した時に、文字化けするだけでなく、文字の組み合わせによってはエラーになってしまう場合があります。

・スペースなのに「不正な"@"があります」といわれた時

スペースの表現に、半角スペースや全角スペースやタブを混在して使用している時、何も問題がないはずなのに、文字コードの関係で「不正な@があります(英文)」というエラーが出る時があります。この問題を回避するため、スペースを空ける時には半角英数のスペースに統一して使用することをお勧めします。また、不要なスペースは、極力削除しておくこともお勧めします。

レイアウト上の問題

・大きめの画像の後にJavaScriptを記述した時

Netscape Navigator2.xでは、大きな画像ファイルやカウンターなどの表示に時間がかかるものがあると、それ以降のJavaScriptが動かない場合があります。この問題は、画像ファイルの前にスクリプトを記述したり、「WIDTH」・「HEIGHT」属性を指定することによって、ほぼ解決されます。

・テーブルの中・テーブルの後ろにJavaScriptを記述した時

テーブル内、あるいはテーブルのすぐ後にJavaScriptを記述した時も、スクリプトが動かない場合があります。この問題は、Netscape Navigator2.x・3.xで見られ、テーブルのサイズに余裕を持たせたり、テーブルより前にスクリプトを記述することによって、ほぼ解決されます。

・アニメーションGIFと同時に使用した時

Netscape Navigator2.xで、アニメーションGIFとJavaScriptを同時に使用しているページを見た場合、ページの読み込みが途中で止まったり、ブラウザが不安定になってシステムエラーなどを引き起こす場合があります。この問題は、Netscape Navigator3.0では解決していますが、それ以前のブラウザを対象にしたページでは、アニメーションGIFとJavaScriptの同時使用は避けたほうがよいでしょう。

「window.open()」を使用する時の注意

・開いたウインドウに文字が記述されない・最後の１行が表示されない

「window.open()」で開いたウインドウに「document.write()」で文字を書き出した時、「document.close()」でドキュメントストリームを明示的に閉じていないと、全く表示されなかったり、最後の1行が表示されなかったりする場合があります。

・window.open()で開いたウインドウ内の画像はフルパスで

「window.open()」で開いたウインドウに画像ファイルを表示させる時、Netscape Navigator2.xなど一部のブラウザでURLをフルパスで指定していないと、画像が表示されない場合があります。

デバッグ時のTIP

・「setTimeout()」は時間設定を大きめにしてテストする

リアルタイムの時間表示やアニメーションするJavaScriptのような、一定時間に処理を繰り返すスクリプト中ににミスがあると、エラーを表示するウインドウが繰り返し出て、どうしようもない状態になる時があります。このようなスクリプトをデバックする時には、繰り返し時間の設定を長めにしておいて、問題がなくなってから設定時間を短くすることをお勧めします。

・ブラウザは１度終了させてから

キャッシュ機能が働くため、ブラウザの「Reload」ボタンを押しただけでは、HTMLファイルが更新されない場合があります。もし、記述ミスや構文ミスが何もないのにまだエラーが出る時には、1度ブラウザを終了させてからファイルを読み込み直してみてください。

JavaScriptのバージョン記述によるエラー回避

<SCRIPT>タグ内の「LANGUAGE」オプション指定で、「LANGUAGE=JavaScript1.1」や「LANGUAGE=JavaScript1.2」といった具合にJavaScriptのバージョンを記述すれば、ページを読み込む時に、そのバージョンのブラウザでは未対応のJavaScriptが記述してあってもエラーになることはありません。

しかし、このままではイベントハンドラでイベントを発生させた時などに、エラーを起こしてしまう場合があります。このような問題を回避するための1つの方法として、ダミーのスクリプトを設定するという方法があります。

例えば、「Imageオブジェクト」の「アニメーションにスタート・ストップボタンを付ける」(P.404)の場合、Imageオブジェクトに未対応のNetscape Navigator2.xのブラウザでこのページこのままを見ると、ページの読み込み時には何も起こらないのですが、「スタート」あるいは「ストップ」ボタンを押した途端、エラー表示のウィンドウが開いてしまいます。

このような場合、次のように画像ファイルをアニメーションさせるスクリプトの前に、何もしないダミーのスクリプトを記述すれば、Netscape Navigator2.xでは「Language」オプションが「JavaScript1.1」のスクリプトには反応しないので、ダミーのスクリプトが評価されてエラー表示のウィンドウは表示されません。

```
<SCRIPT Language="JavaScript">
<!---
function anime1(){ }
function stop1(){ }
//---->
</SCRIPT>
<SCRIPT Language="JavaScript1.1">
<!---
var TimeSet1 = 500 ;
var ImageSetA = 1 ;

  ～
```

また、Netscape Navigator3.0以上のブラウザでは、両方のスクリプトが認識されますが、同じ関数名があった場合、より後に記述された関数が評価されるので、正常にスクリプトが実行されます。

もちろん、ダミーの部分にNetscape Navigator2.x用の処理を設定しても構いません。

本書ではこの他にも、ブラウザーの種類やバージョンによってページを振り分ける(「locationオブジェクト」の「各ブラウザ専用ページに振り分ける」:P.357)などの方法を紹介していますのでそちらも参考にしてください。

Netscape Navigatorのエラーウィンドウ

Netscape Navigator4.06からJavaScript1.3がサポートされたのと共に、今まで問題があるJavaScriptのコードを含んだHTMLファイルをブラウザで読み込んだ時に表示されていた、JavaScript Errorのウィンドウが表示されなくなりました。

この対応は、ネットサーフィンをしている最中、ユーザーは何も悪くないのに突然エラーウィンドウが表示されることがなくなったということなので、ユーザーに対しては適切な対応といえます。

なお、JavaScriptの動作チェックをする時などは、ロケーションバーに「javascript:」と打ち込んで開くコンソールウィンドウに、エラーが発生した時のエラーの発生場所や内容が表示されるので、これでデバックを行うことになります。また、「Netscape Preferences」ファイルに次の1行を追加しておくと、JavaScriptエラーが発生した時、自動的にエラー内容を表示したコンソールウィンドウが開くようになります。

```
user_pref("javascript.console.open_on_error", true);
```

JavaScriptのデバッグを行うことが多く、Errorウィンドウは常に表示されるようになっていて欲しいのであれば、これを設定しておくのもよいでしょう。また、「Netscape Preferences」で、コンソールウィンドウを開くように設定した後も、以下のように、「false」を設定することにより、コンソールウィンドウを開かないようにすることができます。

```
user_pref("javascript.console.open_on_error", false);
```

エラーコンソール

JavaScript1.2で追加・変更された用法

ここでは、JavaScript1.2で追加・変更された用法のうち、本文中で解説しきれなかった事柄について、まとめています。

追加.変更されたArrayオブジェクトの用法

次の用法で配列を作成できるようになりました。

【用法】

```
配列名 = [element0, element1, ..., elementn]
```

サンプルは、Arrayオブジェクトの「配列の要素を文字列にして書き出す」(P.496)のサンプルを、この用法を使って書き直したものです。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
hairetu= ["0番目","1番目","2番目","3番目"];
document.write(hairetu.join("/"));
//--->
</SCRIPT>
```

また、次のサンプルの場合、JavaScript1.2以前では「undefined」となりますが、JavaScript1.2では「1」と値を返すようになりました。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
a = new Array(1)
document.write(a[0]);
//--->
</SCRIPT>
```

変更されたtoStringの値

オブジェクトを文字列に変換する「toString()」メソッドを、JavaScript1.2で使用すると、JavaScript1.2で追加されたオブジェクトや配列の形式で書き出されます。

例えば、下のサンプルの場合、書き出される文字列は、「[object Object]」とならずに「{color:red, size:2.2}」となります。

```
<SCRIPT LANGUAGE="JavaScript1.2">
<!---
mystring = new Object();
mystring.color = "red";
mystring.size = 2.2;
document.write(mystring.toString());
//--->
</SCRIPT>
```

変更されたNumberオブジェクトの用法

下のサンプルのように「Number()」メソッドで数値以外の値を設定すると、Netscape Navigator4.0以前のブラウザではエラーになりますが、JavaScript1.2に対応したNetscape Navigator4.0からは、「NaN」の値を返すようになりました。

```
<SCRIPT LANGUAGE="JavaScript">
<!---
x = Number("one")
document.write(x);
//--->
</SCRIPT>
```

変更されたstringオブジェクトの用法

文字列を分割する「split()」メソッドをJavaScript1.2で使用すると、「文字列.split(" ")」としてスペースで分割するように指定しても、タブ・改行などの特殊記号や、複数のスペースなどが認識されるようになりました。

例えば、下のサンプルをJavaScript1.2で実行した場合、結果は「["Welcome", "To", "My", "Home", "Page"]」となり、それ以前のバージョンのJavaScriptでは、「Welcome,,To, My,,,,Home, Page」となります。

```
mozi1="Welcome  To \nMy    Home \nPage"
a=mozi1.split(" ")
document.write(a)
```

文字列の途中の文字を抜き出す「substring()」メソッドは、JavaScript1.2で使用した時、設定された値を厳密に判断するようになりました。

例えば、次のサンプルをJavaScript1.2で実行した場合、抜き出す先頭の文字の値を0以下の数値に設定したり、抜き出す先頭の文字より抜き出す終了位置の文字を大きな数値に設定した時は、エラーになります。

```
var mozi1 = "Welcome To My Home Page";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.substring(-2,12));
    document.write("<BR>");
    document.write(mozi2.substring(12,8));
```

RegularExpression(正規表現)

JavaScript1.2から、一定パターンの文字・キャラクターを設定し、それを使って検索や置き換えができるようになりました。

パターンの設定は、設定する文字やキャラクターを「/」で括ることによって行います。パターンの認識方法は、例えば「/abc/」と設定した場合、完全に同じ「abc」という文字列でなければパターンが合っていると認識されません。また「/ab*c/」と設定した場合では、「abc」だけでなく"b"の後に複数の"b"があった場合、"abbbc"のような場合もパターンが合っていると認識されます。

パターンの設定方法は、この他にもたくさんあります。

RegularExpression(正規表現)関連では、stringオブジェクトにメソッドが追加されるとともに、パターンの設定を取り扱うRegularExpressionオブジェクトが追加されました。

JavaScript

stringオブジェクトの追加メソッド

【メソッド】

match()	指定したパターンと同じパターンを見つける
replace()	指定したパターンを置き換える
split()	指定したパターンの部分で文字列を分ける

RegularExpressionオブジェクト

【正規表現の作成】

```
正規表現 = /パターンの設定/[i|g|gi]
regexp = new RegExp("パターンの設定", ["i"|"g"|"gi"])
```

・オプション(省略可能)

i	大文字、小文字無視
g	完全一致
gi	大文字、小文字無視して完全一致

付録

【プロパティ】

global	完全一致
ignoreCase	大文字、小文字無視
lastIndex	次のパターンマッチを始める位置
source	マッチさせるパターン
input	($_でも可)検索文字列の設定、変更
multiline	($*でも可)改行コードを無視するかどうか。true(無視しない)、false(無視する)で設定
lastMatch	($&でも可)パターンがマッチした最後の文字
lastParen	($+でも可)パターンがマッチした最後のサブストリング
leftContext	($¥Qでも可)パターンがマッチしたものの直前の文字
rightContext	($'でも可)パターンがマッチしたものの次の文字
$1~$9	パターンがマッチしたものの1部を蓄える(9個まで)

【メソッド】

compile()	ループ文内で使うパターンを作成する
exec()	指定したパターンと同じパターンを見つける
test()	同じパターンがあるかどうかテストする

(javaScript1.3)

toSource()	オブジェクトの値を文字列で返す

基本的なRegularExpressionの用法は、以下の通りです。

次のサンプルでは、stringオブジェクトの「match(パターンの設定)」メソッドを使って、一定のパターンの文字と合った文字を検索しています。

```
<SCRIPT LANGUAGE="JavaScript1.2">
s1 = "NetscapeNavgator4.0";
foun = s1.match(/Nav/);
document.write(foun);
</SCRIPT>
```

次のサンプルでは、stringオブジェクトの「replace(パターンの設定,"入れ替える文字列")」メソッドを使って、指定した文字列を別の文字列に入れ替えています。

```
<SCRIPT LANGUAGE="JavaScript1.2">
s2 = "NetscapeNavgator4.0";
repl = s2.replace(/Navgator/,"Communicat");
document.write(repl);
</SCRIPT>
```

次のサンプルでは、stringオブジェクトの「split(パターンの設定)」メソッドを使って、パターンが合った部分で文字列を分割しています。

```
<SCRIPT LANGUAGE="JavaScript1.2">
s3 = "NetscapeNavgator4.0";
spli = s3.split(/a/);
document.write(spli);
</SCRIPT>
```

次のサンプルでは、「new RegExp("パターンの設定")」を使ってRegularExpressionオブジェクトを作成し、「exec(検索する文字列)」メソッドを使って、一定のパターンの文字と合った文字を検索しています。

stringオブジェクトの「match()」メソッドと似ていますが、この場合はドキュメントがアップデートされます。

```
<SCRIPT LANGUAGE="JavaScript1.2">
re = new RegExp("Nav");
s4 = "NetscapeNavgator4.0";
exe = re.exec(s4);
document.write(exe);
</SCRIPT>
```

Signed Scriptの使い方

JavaScript1.2ではセキュリティなどの関係で、ディスプレイの表示領域外に新しいウインドウを開くことができない、100×100ピクセル以下のサイズのウインドウを新しく開くことができないなど、スクリプトの一部に制約をかけています。

こういった制約を超えてスクリプトを実行したい場合や、スクリプトの動作前にユーザーに認証をしてもらいたい時などに、Java Archive(JAR)を利用して、ユーザーにそのスクリプトの開発元等の情報を提示し、ユーザーの承認を得てからスクリプトを実行させるということができます。この認証手続きの用法を、「Signed Script」といいます。

Signed Scriptは、次の例のように<SCRIPT>タグ内で「ARCHIVE」と「ID」を指定して行い、その内「ARCHIVE」はデジタル認証のソースであるJava Archive(JAR)ファイル名を、「ID」はJava Archive(JAR)ファイルからの著名と組み合わせて使う文字列を指定します。

```
<SCRIPT ARCHIVE="myArchive.jar" ID="1">
document.write("これは、Signed Scriptです。");
</SCRIPT>
```

イベントハンドラと組み合わせて使用する場合は、イベントハンドラが含まれるタグ内には「ARCHIVE」を指定する必要はありませんが、次の例のように、イベントハンドラよりも前にSigned Scriptの指定を行っておく必要があります。

また、Signed Scriptでは、「<A HREF="javascript:...」(「Linkオブジェクト/Anchorオブジェクト」の「リンクをボタンのように使う・1」：P.366)は使えません。

```
<SCRIPT ARCHIVE="myArchive.jar" ID="1">
...
</SCRIPT>
<FORM>
<INPUT TYPE="button" VALUE="クリックして!!" onClick="alert('これはSigned Script
です!!')" ID="2">
</FORM>
```

「ARCHIVE」で指定するJava Archive(JAR)ファイルは、1種類しか使用できません。始めのHTMLファイル中にJava Archive(JAR)ファイルがあった場合、それ以降の「Signed Script」で指定されたスクリプトは、次の例のように「ID」を指定することにより、同じJava Archive(JAR)ファイルを使用するようにします。

この時、HTMLファイルが異なったプラットフォームのサーバー上に別れて置かれているような場合(例えばWindowsとUNIXの間)、問題が発生する可能性があります。

```
<SCRIPT ARCHIVE="myArchive.jar" ID="1">
document.write("これは、Signed Scriptです。");
</SCRIPT>

<SCRIPT ID="2">
document.write("これも、Signed Scriptです。");
document.write("<BR>");
document.write("myArchive.jarのデジタル認証を使用しています。");
</SCRIPT>
```

JARの詳しい情報は、下のサイトから得られます。

・「USigning Software with Netscape Signing Tool.60」
　http://developer.netscape.com/docs/manuals/signedobj/zigbert/index.htm

Live Connect

Netscape Navigator3.0からは、JavaScriptを使ってブラウザ上のJavaアプレットやプラグインを制御するための仕掛けが用意されました。この仕掛けを「Live Connect」といいます。

Live Connectを使用するためには、JavaアプレットもプラグインもLive Connectに対応している必要があります。

Netscape社では、ブラウザ上からWebサーバーの設定を可能にするためにこのLive Connectの技術を使うなど、ブラウザを他の機能と結び付けるための接着剤の役割として使っており、今後ますます利用範囲が広がって行くと思われます。

Live Connectに関したJavaScript側で用意されているオブジェクトには、Appletオブジェクトとembeds配列があります。

Appletオブジェクト(配列)

【用法】

```
・document.applets[インデックス]
・document.applets.length
```

【プロパティ】

length	アプレットの数(配列数)

【メソッド】

JavaAppletのすべての"public method"

eval()	文字列を数値に変える
toString()	オブジェクトを文字列に変える
valueOf()	オブジェクトの値を返す

embeds配列

【用法】

```
・document.embeds[インデックス]
・document.embeds.length
```

【プロパティ】

length	プラグインの数(配列数)

【メソッド】

各プラグイン独自のメソッド

次のサンプルは、Netscape Navigator3.0が標準で持っているプラグイン「LiveAudio」を、Live Connectを使って制御した例です。

「as01.aiff」、「as02.aiff」、「as02.aiff」の3種類の音声ファイルを用意して、通常のプラグインの操作パネルを見えない状態にし、その代わりに、フォームのボタンが押された時に、ボタンで指定した音声ファイルをLiveAudioで音楽の再生をさせる「play(false)」メソッドを使用して、再生させるようにしています。

*Live Connectを使ったプラグインの制御

ボーン!! 拍手!! バキューン!!

```
<HTML>
<HEAD>
<TITLE>LIVE CONNECT</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
function LC(p) { document.embeds[p].play(false) }
//--->
</SCRIPT>
</HEAD>
<BODY>
*Live Connectを使ったプラグインの制御<P>
<EMBED SRC="as01.aiff" HIDDEN=TRUE>
<EMBED SRC="as02.aiff" HIDDEN=TRUE>
<EMBED SRC="as03.aiff" HIDDEN=TRUE>
<P>
<FORM>
 <INPUT TYPE="button" NANE= "LC1" VALUE="ボーン!!" onClick="LC(0)">
 <INPUT TYPE="button" NANE= "LC2" VALUE="拍手!!" onClick="LC(1)">
 <INPUT TYPE="button" NANE= "LC3" VALUE="バキューン!!" onClick="LC(2)">
</FORM>
</BODY>
</HTML>
```

注意 LiveAudioのサポート

68kMacintoshなどのプラットフォームによっては、LiveAudioのLive Connectがサポートされていない場合があります。

JavaScriptの2000年問題

JavaScript1.3では、西暦を4桁で表示する「getFullYear()」メソッドが追加されました。今後は、なるべくこのメソッドを使うことをお勧めします。しかしながら、現時点では、JavaScript1.3に未対応のブラウザが多く出回っているので、「getYear()」メソッドを使わなければいけない場合も多いでしょう。

「getYear()」メソッドは、基本的に西暦の下2桁を取得するメソッドなので、1999年は「99」、2000年になると「100」の値を返します。しかし厳密にいうと、JavaScript1.1では、1999年までは西暦の下2桁を取得し、2000年以降は4桁で取得するように仕様が変更されています。このため、Netscape Navigator3.0などのJavaScript1.1対応ブラウザでは、1999年までは「99」と2桁、2000年になると「getFullYear()」メソッドと同じように、「2000」の値を返します。そして更に複雑なことに、「getFullYear()」メソッドがサポートされてから「getYear()」メソッドは、再び2000年以降も西暦の下2桁を返すように戻されています。

つまり「getYear()」メソッドは、2000年になるとNetscape Navigator2.xでは「100」、Netscape Navigator3.xでは「2000」、Netscape Navigator4.xでは「100」を返すブラウザと「2000」を返すブラウザがあるのです。しかもこの内、Netscape Navigator4.xでは、どのバージョンが「100」を返し、どのバージョンが「2000」を返すかは、OSなどによって違いがあります。

このJavaScriptの2000年にまつわる問題を回避するには、例えば、次のような方法があります。

```
<SCRIPT LANGUAGE="JavaScript">
<!---
now = new Date();
    if ( now.getYear() >= 2000 ){ document.write(now.getYear(),"年") }
    else { document.write(now.getYear()+1900,"年") }
//--->
</SCRIPT>
```

こうしておけば、2000年以降を4桁で返す「getYear()」メソッドはそのまま表示され、それ以外の場合では、取得した値に「1900」を加えることによって、4桁の西暦に直して表示されます。

同じ方法で、「Dateオブジェクト」の「リアルタイムに年・月・日を表示する」(P.456)に、この2000年対策を施すと次のようになります。

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!---
var TC = 0;
function DayWatch() {
    if (TC < 1200) {     //10分後にタイマーを止める処理
    TC++ ;             //10分後にタイマーを止める処理
    var day = new Date();
    if ( day.getYear() >= 2000 ){ var year = day.getYear() }
    else {  var year = day.getYear() +1900 }
    var month = day.getMonth()+1;
    var date = day.getDate();
        if (month < 10) {     //月.日が一桁の時頭に0をつける処理
            month = "0" + month;
                         }
        if (date < 10) {
            date = "0" + date;
                         }
    document.Watch1.watch1.value = year +"."+month+"."+date;
    setTimeout("DayWatch()", 500);
        }                //10分後にタイマーを止める処理
}
//---->
</SCRIPT>
</HEAD>
<BODY  BGCOLOR="#FFFFFF" onLoad="DayWatch()">
*リアルタイムに年.月.日を表示する(2000年対応済)<P>
<FORM NAME="Watch1" >
<INPUT TYPE="text" NAME="watch1" SIZE=10>
</FORM>
</BODY>
</HTML>
```

JavaScript作成ツール

この頃では、Webページ作成用のツールの中に、JavaScriptに対応したものが数多く発表されています。これらのツールのほとんどが、設定画面から項目を選択したり必要な値を設定するだけで、Netscape Navigator・Internet ExplorerどちらでもJavaScriptのコードを自動的に生成することができます。

これらのツールは、自分でスクリプトを書くのと違い、できる範囲はそのツールの機能内に決まっていますが、手軽にJavaScriptを使ったページを作りたいようなユーザーには有効でしょう。また、これらのツールが生成するJavaScriptのコードを眺めるのも、勉強になります。

JavaScriptの自動生成機能を持ったツールをいくつか上げておきますので、参考にしてください。

・「Adobe GoLive」

http://www.adobe.co.jp/product/golive/index.html

アドビシステムズ社のサイト管理機能も含む本格的なWebページ作成・管理ツール。

JavaScriptをページ上に埋め込む機能の他、Netscape Navigator・Internet Explorer両ブラウザに対応したDynamicHTMLを使ったページを作成できる。

・「Dreamweaver」

http://www.macromedia.com/jp/software/dreamweaver/

マクロメディア社のサイト管理機能も含む本格的なWebページ作成・管理ツール。

JavaScriptをページ上に埋め込む機能の他、Netscape Navigator・Internet Explorer両ブラウザに対応したDynamicHTMLを使ったページを作成できる。

・「Adobe ImageReady」

http://www.adobe.co.jp/product/imageready/index.html

アドビシステムズ社のWebページ用の画像編集ツール。

マウスカーソルが画像の上に乗った時に画像を変更するなどの、JavaScriptを使った効果が用意されている。

Adobe ImageReady2.0は、Adobe Photoshop5.5との連携が強められており、両ソフトを切り替えて使用できるようになった。

・「Fireworks」

http://www.macromedia.com/jp/software/fireworks/

マクロメディア社のWebページ用の画像編集ツール。

マウスカーソルが画像の上に乗った時に画像を変更するなどの、JavaScriptを使った効果が用意されている。

・「Painter 5.5 Web Edition」

http://www.metacreations.com/products/painter55/

MetaCreations社の画像編集ツール。

バージョン5.5から、Webページ用の画像編集機能が強化され、マウスカーソルが画像の上に乗った時に画像を変更するなどの、JavaScriptを使った効果が用意されている。

詳解
HTML
&
JavaScript
辞典

索引

目的・機能別索引

数字

アルファベット

あ行

か行

さ行

た行

な行

は行

ま行

や行

ら行

わ行

HTML索引

このページの見方

大文字：要素　小文字：属性　CSS：スタイルシート　アクセシビリティ：アクセシビリティ

記号

A

B

C

D〜E

F

H

I〜K

L

M〜N

N

O

P

Q〜R

S

T

U~W

カナ

JavaScript索引

このページの見方

オブジェクト:オブジェクト　プロパティ:プロパティ　メソッド:メソッド　オプション:オプション
イベントタイプ:イベントタイプ　イベントハンドラ:イベントハンドラ　ビルトイン関数:ビルトイン関数

記号

A～B

Index

JavaScript 索引

C～E

F～G

Index

JavaScript 索引

H～K

L～M

N〜O

P〜R

S

T

U～Z

カナ

Index

JavaScript 索引

ホームページ作成用語索引

数字・記号

A〜D

E〜H

I〜M

N〜Z

あ

い

う

え

お

か

き

く〜け

こ

Index

ホームページ作成用語索引

さ

し

す

せ～そ

た行

な行

は

ひ

ふ

へ～ほ

ま～む

め

も

や行

ら

り

る

れ〜ろ

わ行

ホームページ作成関連リンク

HTML関連リンク

総合

「W3C - The World Wide Web Consortium」

URL http://www.w3.org/

World Wide Webの標準化作業を行っている非営利団体「W3C」のサイト(英語)

「どら猫本舗のリファレンスカウンター」

URL http://www.doraneko.org/

多数のW3Cの仕様を翻訳して公開しているサイト。とても個人の作業とは思えないほどの量とスピードで翻訳されている

「ZSPC」

URL http://www.zspc.com/

HTMLやスタイルシートのリファレンス、DTDの読み方、アクセシビリティ・ガイドラインの日本語訳など、内容豊富なリファレンス・サイト

「とほほのWWW入門」

URL http://wakusei.cplaza.ne.jp/twn/www.htm

HTMLやスタイルシート・JavaScript・Perlなどのリファレンスマニュアル、Dynamic HTML・CGI・SSI・カウンター設置方法など、ページ作成に関する情報満載。誰でも投稿可能なQ&Aコーナーも賑わっている

「NTT DoCoMo Net」

URL http://www.nttdocomo.co.jp/

サイト内にiモード用のホームページ制作に関する詳しい解説がある

HTML

「HTML 4.0 Specification」

URL http://www.w3.org/TR/REC-html40/

HTML4.0の仕様書の原文(英語)

「HTML 3.2 Reference Specification」

URL http://www.w3.org/TR/REC-html32.html

HTML3.2の仕様書の原文(英語)

「W3C HTML Validation Service」

URL http://validator.w3.org/

正しいHTMLになっているかどうかを検証してくれるサービス(英語)

「HTML Help by The Web Design Group」

URL http://www.htmlhelp.com/

ホームページ制作のための正統派リファレンス・サイト。W3C本家の仕様書より分かりやすく、内容もうまくまとめられている(英語/日本語)

CSS

「Cascading Style Sheets, level 1」

URL http://www.w3.org/TR/REC-CSS1
CSS1の仕様書の原文(英語)

「Cascading Style Sheets, level 2」

URL http://www.w3.org/TR/REC-CSS2/
CSS2の仕様書の原文(英語)

「W3C CSS Validation Service」

URL http://jigsaw.w3.org/
正しいスタイルシートになっているかどうかを検証してくれるサービス(英語)

アクセシビリティ

「Web Content Accessibility Guidelines 1.0」

URL http://www.w3.org/TR/WAI-WEBCONTENT/
アクセシビリティ・ガイドライン1.0の原文(英語)

「Web コンテンツ アクセシビリティ・ガイドライン 1.0」

URL http://www.zspc.com/doc/accessibility/
アクセシビリティ・ガイドライン1.0の日本語訳

「こころWeb」

URL http://www.jeida.or.jp/document/kokoroweb/
障害を持つ方々のパソコン利用やコミュニケーションを支援するためのサイト。
具体的な製品や事例などが紹介されている

「株式会社ユーディット(情報のユニバーサルデザイン研究所)」

URL http://www.udit-jp.com/
誰もが使いやすい情報社会のあり方を提案する株式会社ユーディットのサイト。
「ユニバーサルデザインってどんなもの？」・「アクセシビリティって何？」という人には一見の価値あり

JavaScript関連リンク

Netscape社のリファレンス

「JavaScript Developer Central 」

URL http://developer.netscape.com/tech/javascript/index.html
JavaScriptに関する情報を提供するホームページ(英語)

「JavaScript Guide」

URL http://www.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html
JavaScript 1.1のガイド(英語)

「What's New in JavaScript 1.2」

URL http://developer.netscape.com/docs/manuals/communicator/jsguide/js1_2.htm
JavaScript 1.2のガイド(英語)

「What's New in JavaScript 1.3」

URL http://developer.netscape.com/docs/manuals/communicator/jsref/js13.html
JavaScript 1.3のガイド(英語)

「JavaScript Guide」

URL http://developer.netscape.com/docs/manuals/communicator/jsguide4/index.htm
サーバーサイドJavaScriptとクライアントサイドJavaScriptのガイド(英語)

「JavaScript Reference」

URL http://developer.netscape.com/docs/manuals/communicator/jsref/index.htm
サーバーサイドJavaScriptとクライアントサイドJavaScriptのリファレンス(英語)

「Client-Side JavaScript Guide」

URL http://developer.netscape.com/docs/manuals/js/client/jsguide/index.htm
クライアントサイドJavaScriptのガイド(英語)

「Client-Side JavaScript Reference」

URL http://developer.netscape.com/docs/manuals/js/client/jsref/index.htm
クライアントサイドJavaScriptのリファレンス(英語)

Microsoft社のリファレンス

「JScript」

URL http://msdn.microsoft.com/scripting/jscript/default.htm
Microsoft社のJavaScript互換のスクリプトJScriptの解説(英語)

「JScript」

URL http://www.microsoft.com/Japan/Developer/Scripting/JScript/default.htm
JScriptの日本語ページ解説ページ

メーリングリスト

「js-ML」

URL http://www.fureai.or.jp/~tato/js-ml/jsml.htm
Dynamic HTML及びDHTMLcross-browser、JavaScript関連のメーリングリスト

ネットニュース

comp.lang.javascript

英語のJavaScriptの話題を取り扱ったニュースグループ

fj.lang.javascript

日本語のJavaScriptの話題を取り扱ったニュースグループ

その他のページ

「一撃必殺JavaScript日本語リファレンス」

URL http://www.shiojiri.ne.jp/~openspc/
情報量は随一

「独学JavaScript」

URL http://www.ueda.info.waseda.ac.jp/~gaku/js/index.html
JavaScript関連Pageの老舗

「JavaScriptTips集」

URL http://www.din.or.jp/~hagi3/JavaScript/JSTips/Default.htm
JavaScript、DHTML関連、cross-browserDHTML情報多数

「WebPageで使えるJavaScript」

URL http://www.bekkoame.ne.jp/~hamba/webimage/java/java.html
JavaScript関連リンクが充実している

ホームページ作成参考書籍

HTML関連書籍

「はじめてのSGML」(技術評論社)
編著：(株)日本ユニテックSGMLサロン

「HTMLハンドブック」(ソフトバンク)
著者：渡辺竜生

「詳解HTML4.0&Cascading Style Sheet辞典」(秀和システム)
著者：岡蔵龍一

「詳解HTML&JavaScript辞典」(秀和システム)
著者：岡崎桂子,長谷川豊,半場方人

JavaScript関連書籍

「だれでもカンタンJavaScriptサンプル集」(秀和システム)
著者：高橋登史郎

「JavaScript入門」(技術評論社)
著者：半場方人

「ちょ～初心者のためのJavaScript入門」(秀和システム)
著者：半場方人

「改定第2版JavaScriptポケットリファレンス」(技術評論社)
著者：古籏一浩

「OFFICIAL Netscape JavaScript」(BNN)
著者：PeterKent,JohnKent

「詳解JavaScript&DynamicHTML辞典」(秀和システム)
著者：半場方人

DynamicHTML関連書籍

「ダイナミックHTMLファーストステップ」(秀和システム)
著者：松尾忠則

「だれでもカンタンDynamicHTMLサンプル集」(秀和システム)
著者：高橋登史郎

「DynamicHTMLでつくるホームページ」(技術評論社)
監修：古籏一浩　著者：松尾忠則,半場方人,すぎうらしろう

正式な文書を作るうえで必要なHTML4.0と、自由度の高いレイアウトを実現するCascading Style Sheet2を、辞書形式にまとめたWebデザイナー必携の書!!

詳解HTML&Cascading Style Sheet辞典

岡蔵龍一/著　定価(本体2,000円+税)

ISBN4-87966-828-1

DynamicHTMLを作る基本として必要なJavaScript1.2と、そのまま使用できる簡単で短いDynamicHTMLのサンプルを、辞書形式にまとめたWebデザイナー必携の書!!

詳解JavaScript&DynamicHTML辞典

半場方人/著　定価(本体2,200円+税)

ISBN4-87966-829-X

サンプルをコピーして使うことで理解を早めたり、作成時間を短縮できます。100以上のサンプルを収録したCD-ROMを解説と共に利用すれば、簡単に動くホームページが作れます。

だれでもカンタンJavaScriptサンプル集

高橋登史朗/著　定価（本体2,800円+税）

ISBN4-87966-706-4

CD-ROMに収録されたサンプルを見開き展開の解説に沿って改変・設置すれば、今日からあなたのホームページでアクセスカウンタやチャットルームが使えます。

だれでもカンタンCGI&SSIサンプル集

古川剛・谷中一朝・成田政司/著　定価（本体2,800円+税）

ISBN4-87966-774-9

サンプルを中心としたダイナミックHTMLの入門書。サンプルソースをコピーして、HTMLファイルにペーストするだけで、動きのあるホームページを作成できます。

だれでもカンタンJavaScriptサンプル集

高橋登史朗/著　定価（本体2,800円+税）

ISBN4-87966-807-9

「動くアクセスカウンタ」・「Webページに埋め込める掲示板」などの実用的なプログラムを満載した、CGI・SSI・JavaScriptだけでは実現できない、JAVAならではのサンプル集です。

だれでもカンタンJAVAアプレットサンプル集

Digital Cat LLC.・佐藤治/著　定価（本体2,800円+税）

ISBN4-87966-808-7

スクリプティング未経験の超初心者でも抵抗感のない、新しいタイプの入門書。最新バージョンである「JavaScript1.3」にも対応しています。

ちょー初心者のためのJavaScript入門

半場方人/著　定価(本体2,200円+税)

ISBN4-87966-706-4

(株)秀和システムのホームページ

http://www.shuwasystem.co.jp/

新刊書籍のご案内などの最新情報や既刊書籍の検索、サポートページへのリンクなどがございます。
また、直接購入のご案内もさせていただいております。

本書のサポートページ

URL ： http://www.shuwasystem.co.jp/~book0931 /

パスワード ： OHSUNM

本書で紹介しているサンプルの閲覧やダウンロードができます。
パスワードは、半角英大文字で入力してください。

カバーデザイン・本文デザイン協力
株式会社アサヒ・エディグラフィ（成田　英夫）

最新カラー版
詳解HTML&JavaScript辞典

発行日	1999/10/8	初版第1刷 発行
	2000/11/15	初版第5刷 発行

著　者　岡蔵　龍一 / 半場　方人

発行者　牧谷　秀昭
発行所　株式会社 秀和システム
〒107-0062　東京都港区南青山1-26-1寿光ビル5F
FAX 03-3405-7538
印刷所　岩岡印刷工業株式会社
© Ryuichi Okakura/Masahito Hamba

Printed in Japan

ISBN4-87966-931-8 C3055
定価はカバーに表示してあります。
落丁本、乱丁本はお取り替えいたします。

●本書の内容に関するお問い合わせ先
住所、氏名、電話番号を明記の上、書面にてお送りください。

カラーチャート1：HTML4.0で名前が定義されている色

HTML4.0でcolor属性の値として定義されている色名は、以下の通りです(色の名前には大文字と小文字の区別はありません)。

これらは、CSS2でも同様にプロパティの値として利用できます。色を指定する属性はHTMLの次のバージョンで廃止されることになっており、代わりにスタイルシートを使用することが推奨されています。

具体的な色の指定の方法は、HTMLの場合は「HTMLについて」の「色の指定方法」(P.25)、CSSの場合は「スタイルシートについて」の「色の指定方法」(P.187)を参照してください。

#000000	Black	#008000	Green
#C0C0C0	Silver	#00FF00	Lime
#808080	Gray	#808000	Olive
#FFFFFF	White	#FFFF00	Yellow
#800000	Maroon	#000080	Navy
#FF0000	Red	#0000FF	Blue
#800080	Purple	#008080	Teal
#FF00FF	Fuchsia	#00FFFF	Aqua

Web Safeカラーについて

パソコンの画面上の色は、RGB(Red、Green、Blue)の色をそれぞれ0～255までの256段階に調節して色を表現しています。したがって、フルカラーでは256×256×256(R×G×B)の16,777,216色が表現されます。

「Web Safe カラー」とは、この各256段階あるRGB値を6段階にしたものです。そして、その6段階の値とは、0・51・102・153・204・255と、0から51ずつ増やしていった数になっています。51は16進数でいうと「33」、割合でいうと「20%」にあたる数です。つまり、「Web Safeカラー」は、RGBの各値(画面上では光の強さ)を0%から20%ずつ上げていった値の組み合せで作られているということになります。

10進数	0	51	102	153	204	255
16進数	0	33	66	99	CC	FF
%	0%	20%	40%	60%	80%	100%

これらを組み合わせると、RGBの3色がそれぞれ6段階あるので、6×6×6の216色がWeb Safe カラーということになります。

具体的な色については、次ページを参照してください。

カラーチャート2：Web Safeカラー

WindowsやMacintoshなどのプラットフォームでそれぞれ設定されている、256色のシステムカラーパレットのうち、共通している216色のカラーチャートです。

これらの色を使うことで、異なるOS環境でも色化けなどを生じさせることなく表示することができます。

印刷された色は、実際に画面に表示されている色とは多少異なります。大体の色の感じを掴むためにご利用ください。

#000000 (000,000,000)	#003300 (000,051,000)	#006600 (000,102,000)	#009900 (000,153,000)	#00CC00 (000,204,000)	#00FF00 (000,255,000)
#000033 (000,000,051)	#003333 (000,051,051)	#006633 (000,102,051)	#009933 (000,153,051)	#00CC33 (000,204,051)	#00FF33 (000,255,051)
#000066 (000,000,102)	#003366 (000,051,102)	#006666 (000,102,102)	#009966 (000,153,102)	#00CC66 (000,204,102)	#00FF66 (000,255,102)
#000099 (000,000,153)	#003399 (000,051,153)	#006699 (000,102,153)	#009999 (000,153,153)	#00CC99 (000,204,153)	#00FF99 (000,255,153)
#0000CC (000,000,204)	#0033CC (000,051,204)	#0066CC (000,102,204)	#0099CC (000,153,204)	#00CCCC (000,204,204)	#00FFCC (000,255,204)
#0000FF (000,000,255)	#0033FF (000,051,255)	#0066FF (000,102,255)	#0099FF (000,153,255)	#00CCFF (000,204,255)	#00FFFF (000,255,255)
#330000 (051,000,000)	#333300 (051,051,000)	#336600 (051,102,000)	#339900 (051,204,000)	#33CC00 (051,204,000)	#33FF00 (051,255,000)
#330033 (051,000,051)	#333333 (051,051,051)	#336633 (051,102,051)	#339933 (051,153,051)	#33CC33 (051,204,051)	#33FF33 (051,255,051)
#330066 (051,000,102)	#333366 (051,051,102)	#336666 (051,102,102)	#339966 (051,153,102)	#33CC66 (051,204,102)	#33FF66 (051,255,102)
#330099 (051,000,153)	#333399 (051,051,153)	#336699 (051,102,153)	#339999 (051,153,153)	#33CC99 (051,204,153)	#33FF99 (051,255,153)
#3300CC (051,000,204)	#3333CC (051,051,204)	#3366CC (051,102,204)	#3399CC (051,153,204)	#33CCCC (051,204,204)	#33FFCC (051,255,204)
#3300FF (051,000,255)	#3333FF (051,051,255)	#3366FF (051,102,255)	#3399FF (051,153,255)	#33CCFF (051,204,255)	#33FFFF (051,255,255)
#660000 (102,000,000)	#663300 (102,051,000)	#666600 (102,102,000)	#669900 (102,153,000)	#66CC00 (102,204,000)	#66FF00 (102,255,000)
#660033 (102,000,051)	#663333 (102,051,051)	#666633 (102,102,051)	#669933 (102,153,051)	#66CC33 (102,204,051)	#66FF33 (102,255,051)
#660066 (102,000,102)	#663366 (102,051,102)	#666666 (102,102,102)	#669966 (102,153,102)	#66CC66 (102,204,102)	#66FF66 (102,255,102)

#660099 (102,000,153)	#663399 (102,051,153)	#666699 (102,102,153)	#669999 (102,153,153)	#66CC99 (102,204,153)	#66FF99 (102,255,153)
#6600CC (102,000,204)	#6633CC (102,051,204)	#6666CC (102,102,204)	#6699CC (102,153,204)	#66CCCC (102,204,204)	#66FFCC (102,255,204)
#6600FF (102,000,255)	#6633FF (102,051,255)	#6666FF (102,102,255)	#6699FF (102,153,255)	#66CCFF (102,204,255)	#66FFFF (102,255,255)
#990000 (153,000,000)	#993300 (153,051,000)	#996600 (153,102,000)	#999900 (153,153,000)	#99CC00 (153,204,000)	#99FF00 (153,255,000)
#990033 (153,000,051)	#993333 (153,051,051)	#996633 (153,102,051)	#999933 (153,153,051)	#99CC33 (153,204,051)	#99FF33 (153,255,051)
#990066 (153,000,102)	#993366 (153,051,102)	#996666 (153,102,102)	#999966 (153,153,102)	#99CC66 (153,204,102)	#99FF66 (153,255,102)
#990099 (153,000,153)	#993399 (153,051,153)	#996699 (153,102,153)	#999999 (153,153,153)	#99CC99 (153,204,153)	#99FF99 (153,255,153)
#9900CC (153,000,204)	#9933CC (153,051,204)	#9966CC (153,102,204)	#9999CC (153,153,204)	#99CCCC (153,204,204)	#99FFCC (153,255,204)
#9900FF (153,000,255)	#9933FF (153,051,255)	#9966FF (153,102,255)	#9999FF (153,153,255)	#99CCFF (153,204,255)	#99FFFF (153,255,255)
#CC0000 (204,000,000)	#CC3300 (204,051,000)	#CC6600 (204,102,000)	#CC9900 (204,153,000)	#CCCC00 (204,204,000)	#CCFF00 (204,255,000)
#CC0033 (204,000,051)	#CC3333 (204,051,051)	#CC6633 (204,102,051)	#CC9933 (204,153,051)	#CCCC33 (204,204,051)	#CCFF33 (204,255,051)
#CC0066 (204,000,102)	#CC3366 (204,051,102)	#CC6666 (204,102,102)	#CC9966 (204,153,102)	#CCCC66 (204,204,102)	#CCFF66 (204,255,102)
#CC0099 (204,000,153)	#CC3399 (204,051,153)	#CC6699 (204,102,153)	#CC9999 (204,153,153)	#CCCC99 (204,204,153)	#CCFF99 (204,255,153)
#CC00CC (204,000,204)	#CC33CC (204,051,204)	#CC66CC (204,102,204)	#CC99CC (204,153,204)	#CCCCCC (204,204,204)	#CCFFCC (204,255,204)
#CC00FF (204,000,255)	#CC33FF (204,051,255)	#CC66FF (204,102,255)	#CC99FF (204,153,255)	#CCCCFF (204,204,255)	#CCFFFF (204,255,255)
#FF0000 (255,000,000)	#FF3300 (255,051,000)	#FF6600 (255,102,000)	#FF9900 (255,153,000)	#FFCC00 (255,204,000)	#FFFF00 (255,255,000)
#FF0033 (255,000,051)	#FF3333 (255,051,051)	#FF6633 (255,102,051)	#FF9933 (255,153,051)	#FFCC33 (255,204,051)	#FFFF33 (255,255,051)
#FF0066 (255,000,102)	#FF3366 (255,051,102)	#FF6666 (255,102,102)	#FF9966 (255,153,102)	#FFCC66 (255,204,102)	#FFFF66 (255,255,102)
#FF0099 (255,000,153)	#FF3399 (255,051,153)	#FF6699 (255,102,153)	#FF9999 (255,153,153)	#FFCC99 (255,204,153)	#FFFF99 (255,255,153)
#FF00CC (255,000,204)	#FF33CC (255,051,204)	#FF66CC (255,102,204)	#FF99CC (255,153,204)	#FFCCCC (255,204,204)	#FFFFCC (255,255,204)
#FF00FF (255,000,255)	#FF33FF (255,051,255)	#FF66FF (255,102,255)	#FF99FF (255,153,255)	#FFCCFF (255,204,255)	#FFFFFF (255,255,255)

カラーチャート3：Color Name

HTML4.0・CSSで正式に設定されている色、プラットフォーム間で共通な色、というように紹介してきましたが、最後はブラウザで設定されている色です。

以下のチャートは、Internet ExplorerとNetscape Navigatorで名前が定義されている140色です(HTML4.0で定義されている16色も含まれています)。色を名前で設定する時に利用することができますが、あくまでもブラウザが対応しているものであるので、不安定な部分があることに注意してください。

印刷された色は、実際に画面に表示されている色とは多少異なります。大体の色の感じを掴むためにご利用ください。

Code	Name
#FFFFFF	White
#FFFAFA	Snow
#F8F8FF	GhostWhite
#F5F5F5	WhiteSmoke
#FFFAF0	FloralWhite
#FAF0E6	Linen
#FAEBD7	AntiqueWhite
#FFEFD5	PapayaWhip
#FFEBCD	BlanchedAlmond
#FFE4C4	Bisque
#FFE4B5	Moccasin
#FFDEAD	NavajoWhite
#FFDAB9	PeachPuff
#FFE4E1	MistyRose
#FFF0F5	LavenderBlush
#FFF5EE	Seashell
#FDF5E6	OldLace
#FFFFF0	Ivory
#F0FFF0	Honeydew
#F5FFFA	MintCream
#F0FFFF	Azure
#F0F8FF	AliceBlue
#E6E6FA	Lavender
#000000	Black
#2F4F4F	DarkSlateGray
#696969	DimGray
#808080	Gray
#A9A9A9	Darkgray

Code	Name
#C0C0C0	Silver
#D3D3D3	LightGrey
#DCDCDC	Gainsboro
#778899	LightSlateGray
#708090	SlateGray
#B0C4DE	LightSteelBlue
#4682B4	SteelBlue
#4169E1	RoyalBlue
#191970	MidnightBlue
#000080	Navy
#00008B	Darkblue
#0000CD	MediumBlue
#0000FF	Blue
#1E90FF	DodgerBlue
#6495ED	CornflowerBlue
#00BFFF	DeepSkyBlue
#87CEFA	LightSkyBlue
#87CEEB	SkyBlue
#ADD8E6	LightBlue
#B0E0E6	PowderBlue
#AFEEEE	PaleTurquoise
#E0FFFF	LightCyan
#00FFFF	Cyan
#00FFFF	Aqua
#40E0D0	Turquoise
#48D1CC	MediumTurquoise
#00CED1	DarkTurquoise
#20B2AA	LightSeaGreen

Code	Name
#5F9EA0	CadetBlue
#008B8B	Darkcyan
#008080	Teal
#2E8B57	SeaGreen
#556B2F	DarkOliveGreen
#006400	DarkGreen
#008000	Green
#228B22	ForestGreen
#3CB371	MediumSeaGreen
#8FBC8F	DarkSeaGreen
#66CDAA	MediumAquamarine
#7FFFD4	Aquamarine
#98FB98	PaleGreen
#90EE90	LightGreen
#00FF7F	SpringGreen
#00FA9A	MediumSpringGreen
#7CFC00	LawnGreen
#7FFF00	Chartreuse
#ADFF2F	GreenYellow
#00FF00	Lime
#32CD32	LimeGreen
#9ACD32	YellowGreen
#6B8E23	OliveDrab
#808000	Olive
#BDB76B	DarkKhaki
#EEE8AA	PaleGoldenrod
#FFF8DC	Cornsilk
#F5F5DC	Beige
#FFFFE0	LightYellow
#FAFAD2	Lightgoldenrodyellow
#FFFACD	LemonChiffon
#F5DEB3	Wheat
#DEB887	Burlywood
#D2B48C	Tan
#F0E68C	Khaki
#FFFF00	Yellow
#FFD700	Gold
#FFA500	Orange
#F4A460	SandyBrown
#FF8C00	DarkOrange
#DAA520	Goldenrod
#CD853F	Peru
#B8860B	DarkGoldenrod
#D2691E	Chocolate
#A0522D	Sienna
#8B4513	SaddleBrown
#800000	Maroon
#8B0000	DarkRed
#A52A2A	Brown
#B22222	Firebrick
#CD5C5C	IndianRed
#BC8F8F	RosyBrown
#E9967A	DarkSalmon
#F08080	LightCoral
#FA8072	Salmon
#FFA07A	LightSalmon
#FF7F50	Coral
#FF6347	Tomato
#FF4500	OrangeRed
#FF0000	Red
#DC143C	Crimson
#C71585	MediumVioletRed
#FF1493	DeepPink
#FF69B4	HotPink
#DB7093	PaleVioletRed
#FFC0CB	Pink
#FFB6C1	LightPink
#D8BFD8	Thistle
#FF00FF	Magenta
#FF00FF	Fuchsia
#EE82EE	Violet
#DDA0DD	Plum
#DA70D6	Orchid
#BA55D3	MediumOrchid
#9932CC	DarkOrchid
#9400D3	DarkViolet
#8B008B	DarkMagenta
#800080	Purple
#4B0082	Indigo
#483D8B	DarkSlateBlue
#8A2BE2	BlueViolet
#9370DB	MediumPurple
#6A5ACD	SlateBlue
#7B68EE	MediumSlateBlue

フォント表示見本

Windows

MSゴシック

日本語漢字見本
あいうえおアイウエオ
0123456789 ！ ” ＃ ＄ ％ ＆ ／ ＊ － ＋

MS Pゴシック

日本語漢字見本
あいうえおアイウエオ
0123456789！”＃＄％＆／＊－＋

MS 明朝

日本語漢字見本
あいうえおアイウエオ
0123456789 ！ ” ＃ ＄ ％ ＆ ／ ＊ － ＋

MS P明朝

日本語漢字見本
あいうえおアイウエオ
0123456789！”＃＄％＆／＊－＋

Arial

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz<
0123456789!"#$%&/*-+

Century Gothic

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Comic Sans MS

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Copperplate

ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789!"#$%&/*-+

Courier

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

Courier New

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

Impact

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Lucida

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/-+*

News Gothic

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Symbol

ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ
αβχδεφγηιφκλμνοπθρστυϖωξψζ
0123456789!∀#∃%&/∗−+

Times New Roman

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Verdana

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

WindowsとMacintoshで最初からOSにインストールされているフォントをブラウザで表示させました。フォント名を指定する時の参考にしてください。

Macintosh

Osaka

日本語漢字見本
あいうえおアイウエオ
0123456789!"#$%&/*-+

Osaka-等幅

日本語漢字見本
あいうえおアイウエオ
0123456789 ! " # $ % & / * - +

細明朝体

日本語漢字見本
あいうえおアイウエオ
0123456789!"#$%&/*-+

中ゴシック体

日本語漢字見本
あいうえおアイウエオ
0123456789!"#$%&/*-+

平成角ゴシック

日本語漢字見本
あいうえおアイウエオ
0123456789!"#$%&/*-+

平成明朝

日本語漢字見本
あいうえおアイウエオ
0123456789!"#$%&/*-+

Charcoal

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Chicago

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Courier

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

Geneva

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Helvetica

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Monaco

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

New York

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Palatino

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Symbol

ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ
αβχδεφγηιφκλμνοπθρστυϖωξψζ
0123456789!∀#∃%&/*-+

Times

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789!"#$%&/*-+

Webサイズチャート

HTML・CSS・JavaScriptでサイズを指定する時に使う単位を解説します。ただし、px・%以外は、スタイルシート内でしか使うことができません。

相対的な大きさの単位

- **px**　1ピクセルを1とする単位です。主にコンピュータのディスプレイを出力対象としますが、その解像度によって表示される大きさが異なってきますので、注意してください。
- **%**　他の大きさに対する割合で表す場合に使用します。対象とする大きさは状況に応じて、同じ要素内の他のプロパティ、親要素のプロパティ、それを含むブロックの幅などになります。
- **em**　その範囲で有効なフォントの高さ(font-sizeの値)を1とする単位です。スタイルシートのfont-sizeプロパティの値としてこの単位が使用された場合には、親要素で有効なフォントの高さを1とする値になります。
- **ex**　その範囲で有効なフォントのアルファベットの小文字「x」の高さを1とする単位です。フォントによっては、必ずしも「x」の高さと一致するとは限りません。また、この単位は「x」を含まないフォントにも適用されます。

em x-height ex

絶対的な大きさの単位

- **in**　インチ。1インチは2.54cm(=25.4mm)です。
- **cm**　センチメートル。1センチメートルは10ミリメートルです。
- **mm**　ミリメートル。1ミリメートルは1/10センチメートルです。
- **pt**　ポイント。1ポイントは1/72インチです。一般的に文字のサイズを指定する時に使います。
- **pc**　パイカ。1パイカは12ポイントです。

【メジャー】

in　0　0.5　1　1.5　2　2.5　3　3.5

cm　0　1　2　3　4　5　6　7　8　9　10

mm　0　5　10　15　20　25　30　35　40　45　50　55　60　65　70　75　80　85　90　95　100

pt　0　50　100　150　200　250

pc　0　5　10　15　20

【文字サイズ】

8pt　あいうえおABCabc1234567890

12pt　あいうえおABCabc1234567890

18pt　あいうえおABCabc1234567890

24pt　あいうえおABCabc123456789

ISBN4-87966-931-8

C3055 ¥2500E

9784879669315

1923055025004

定価　(本体 2500円+税)

最新実用カラー版

詳解 HTML & JavaScript 辞典

岡蔵龍一　半場方人　著

株式会社 秀和システム

秀和システム